HOVIS Lite

User GUIDE

DRC
Mode
OK
CAUTION
DO NOT REMOVE THIS LABEL
HerkuleX
DRS-0101
Dongbu Robot

Dongbu Robot

Dongbu Robot

Hovis Lite User Guide
DR-Visual Logic ロボットプログラミング

初版発行日 ：2011年11月18日
発行 ：(株)Dongbu Robot
住所 ：11th Floor, Bucheon Techno Park 401, Yakdae-dong,
Wonmi-gu, Bucheon-city, Gyunggi-do 420-734, Korea
TEL ：+82-32-329-5551
FAX ：+82-32-329-5569

この教材はDongbu Robotで書かれた教育制作物として事前の承認なしに
無断でコピー、複製、転載することを禁じます。

(株)Dongbu Robot All rights reserved

PART

01

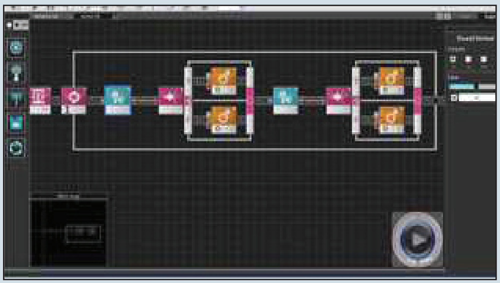
紹介

Hovis Lite 紹介

ホビスライトをご紹介します。

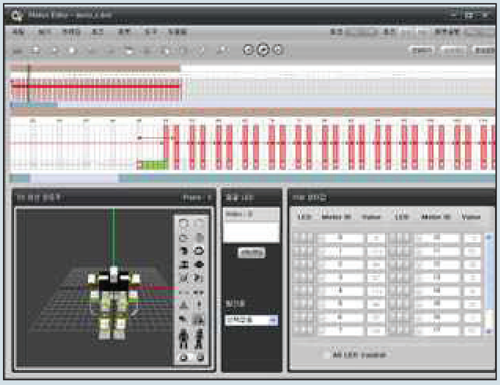
カラフルなブラケット

4色のブラケットを利用してカラフルなヒューマノイドの製作が可能です。また、ケースを被せることで完成形ロボットとオムニホイールの装着、Android端末の装着などのアップグレードが可能な世界初のロボットです。



DR–Visual Logic (Task Editor)

DRCコントローラを基盤にしてプログラミングできるように、全24種類のモジュールを提供します。Drag＆Drop方式でプログラミング言語を全く知らない初心者でも簡単にプログラミングできるグラフィックスのプログラミング言語ツールです。



DR–SIM (Motion Editor)

ロボットのモーションエディタにビデオエディタで使用するタイムフレームを導入し、画面上でモーションを作ることができ、ロボットからのキャプチャも可能な多機能のロボットモーション製作ツールです。作成したモーションは画面のシミュレーションを使用してプレビューができ、ロボットにダウンロードしてすぐに実行することができます。

- 4つの色のブラケット
- 27種類のロボットを組立可能
- ケースごとにアップグレード可能
- Android端末とのプログラミングインターフェイスの提供
- ソースの提供
- カリキュラムの提供

関連サイト

プログラム,ソース,マニュアル,組み立てマニュアルダウンロード :
www.hovis.co.kr/guide
www.dongburobot.com → English → Customer support → For Services → Archives
www.dongburobot.com → Quick Menu → For Service → Archives

警告/注意

以下に表示されている安全に関する注意事項は製品を安全に正しく使用して予期せぬ危険や損害を事前に防止するためのものです。

注意事項は「警告」と「注意」の二つに分かれており、それぞれの意味は次のとおりです。
△警告：その指示に違反した場合、使用者が死亡または重傷を負う可能性があります。
△注意：その指示に違反した場合、ユーザーのけがや物的被害が発生する可能性があります。

警告

・本製品に付属の充電器や本製品に適合するように設計された充電器以外の他の製品は使用しないでください。
過電圧により製品が損傷したり、感電、火災、爆発の危険性があります。
・旅行者のための電圧変換器とアダプタ、マルチプラグなどの変圧装置を利用して充電器に接続しないでください。
過熱による火災や製品損傷の原因となります。
・一つのコンセントに複数の電源プラグを接続しないでください。
異常な発熱や火災の原因になります。
・濡れた手で充電器に触れないでください。
火災や感電の危険性があります。
・電源コードは鋭い物体や子供、ペットの手の届かないようにしてください。
電源コードが曲がったり、他の物に押されないようにし、電源コードに足がつまずかないように設置してください。
火災や感電、深刻な傷害の原因になります。
・充電器の使用時には必ず電源コードやプラグ、その他の部位に損傷がないことを確認の上、破損しているところやひどく磨耗したところがある場合は使用を中止し、新しい充電器と交換してください。
火災や感電、製品損傷、深刻な傷害の原因になります。
・製品を充電している時、充電器は机の上や床などの風通しがよいところに置いてください。通気を妨害するもので覆われていないことに注意してください。ケースなどの密閉された場所に入れたまま使用しないでください。
・充電器の電源コードはコンセントに確実に密着させてください。
火災や感電の原因となります。
・充電器をコンセントに差し込む前に充電器に表記された定格電圧や周波数がコンセントの電源に一致していることを確認してください。
火災や製品損傷の原因となります。
・コンセントに充電器のプラグを差し込む時または取り外す時はプラグのピンの部分に指が触れないようにしてください。
火災や感電の原因となります。
・床に電線や物が煩わしく置かれている場合、ロボットが動くのに邪魔になることがありますので片づけて下さい。
火災や製品損傷の原因となります。
・電池は子供が触れないようにしてください。

警告/注意

- 雷が予想される場合は充電器をコンセントから分離して置き、ロボットとも分離して置いてください。ただし、雷が発生した場合、電源コードに触れることは危険ですので絶対に触れずにロボットを充電器に接続しないでください。
  火災や感電による重傷の原因になります。
- 3歳未満の乳児には使用できません。
- 3歳未満の子供のそばでロボットを使用しないでください。
  人体の傷害や製品の損傷の原因となります。
- 製品は指定されたサービス担当者以外は絶対に分解したり、改造しないでください。
  （取扱説明書の裏面のサービス受付/ご相談の連絡先にお問い合わせください。）
  製品の故障の原因となります。弊社で指定されたサービス担当者以外の者が修理して故障した場合、無償保証修理期間中でも有償サービスとして処理されます。
- ロボットの充電接続端子や充電器の接続端子に電気を通す物を接触させないでください。
  火災や感電の原因となります
- バッテリー、充電器、ロボットを49℃（120℉）を超える高温や、ラジエータやストーブ、電気ヒーター、直射日光、密閉された車中などの熱源にさらさないでください。
  バッテリーの性能の低下や製品の破損、爆発、火災の原因となります。
- テーブルの上や階段などの製品が落ちる危険性があるところと落下物や振動の危険性があるところ、床が堅くないところでロボットを使用しないでください。
  人体損傷や製品の破損の原因となります。
- ロボットの充電の接続端子を針など電気を通すもので突かないでください。
  火災や感電の原因となります。
- 指やその他の異物がロボットの関節や接続部分に挟んだり入らないようにしてください。
  けが、火災や製品損傷の原因となります。
- ロボットの上に座ったり、足で踏んだり、他の物を置かないでください。
  けが、火災や製品損傷の原因となります。
- 製品に水やその他の液体、砂をかけたり、入浴させないでください。
  感電や故障の危険性があります。
- 風呂、洗面台、スイミングプール、濡れた床など湿気の多い環境で製品を使用しないでください。
  感電または製品損傷の原因となります。
- 寒い場所から暖かい場所に製品を移動した場合、製品の表面や内部に結露（水滴、露）が生じることがあります。
  この場合、ロボットの電源が入っているときは電源をオフにして、ロボットを新たに移動した場所で一時間以上そのままにして水分が蒸発することができるようにした後、使用してください。
- ロボットの外側を清潔にするときは乾いた柔らかい巾をご利用ください。
  濡れた布は感電やけがや製品の破損の原因となります。荒くて硬い布は製品の表面に損傷を与えます。
- バッテリーの交換をご希望の場合は本マニュアルの最後のページにあるサービスの受付/相談の問い合わせ先にご連絡ください。本製品はリサイクル可能なリチウムイオンバッテリーを使用しており、廃棄の際にはリサイクルの分別の指示に従って処理する必要があります。専用の電池以外の電池は使用することはできません。

警告/注意

注意

- ボックスを開封してロボットを取り出す時に胴体の下に手を入れて体幹の全体をつかんだ後、持ち上げます。
- 腕、脚、首などの製品の一部だけを持って持ち上げずに、胴全体をつかんで持ち上げてください。
  製品の破損の原因となります。
- 移動や長期間の保管時には背面の主電源スイッチを切ってください。
  バッテリを放電することができます
- バッテリを使用しなくても時間が経てば自然に放電されています。
  再びご使用になる前にバッテリーを充電してからご使用ください。
- ロボットに衝撃を加えたり、落とさないでください。
  破損や故障の原因となります。
- ロボットが動いているときはタッチやリモコンの命令に反応しない場合がありますので無理に操作しないでください。
  ロボットがしていた動作を終えた後にもう一度タッチするかリモコンの命令ボタンを押してください。
- 床が平らでない場合や傾斜、滑りやすいところや障害物が散在しているところではロボットがつまずいて倒れたり、前に進めないことがありますのでご注意ください。
- 磁石や磁気のある物体またはX-線があるところにロボットを置かないでください。
  製品の破損や誤動作の原因となります。
- ロボットは屋内で使用してください。コンクリートの床などの粗い表面では使用しないでください。
  製品の破損の原因となります。
- 移動や長期間の保管時にはロボットや部品などが破損しないように販売時に提供された梱包材や適切な梱包材を使用してください。
- ロボットの関節や動作部位に潤滑剤を注入しないでください。

目次

PART 01

紹介

Hovis Liteの紹介 -------------------- 04
警告および注意 -------------------- 05
組立からテストまで -------------------- 10

PART 02

コントローラー

概要 -------------------- 11
DRCバッテリの取り付け方法 -------------------- 12
DRCの接続方法 -------------------- 13
DRCのインターフェイス -------------------- 14
DRC Register Protocol -------------------- 15
DRC機能の使い方 -------------------- 16
DRC基本テスト -------------------- 18
サーボモーター -------------------- 20
光センサ -------------------- 21
音センサ -------------------- 22
距離センサ(アナログ、デジタル) -------------------- 23
ジャイロセンサ -------------------- 24

PART 03

部品リスト

Hovis Lite構成図-------------------- 25
Hovis Lite部品リスト-------------------- 26

PART 04

組立図

Humanoid 組立図 -------------------- 29
Humanoid DRC 機能の使い方 -------------------- 31

PART 05

ソフトウェア : DR-SIM

- インストール 35
- Hello DR-SIM 39
- ユーザーインターフェイス 43
- ヘルプの使い方 44
- ダウンロード 45
- モーションの作成 46

PART 06

ソフトウェア : DR-Visual Logic

- インストール 58
- Hello DR-Visual Logic 62
- ユーザーインターフェイス 67
- ヘルプの使い方 68
- プログラミングのモジュール 69
- プロパティー 76
- コンパイル/ダウンロード 77
- モジュール別プログラミング 78
  - Move 79
  - Motor 85
  - LED Button 104
  - Light 123
  - Sound 130
  - Sound2 139
  - digital 150
  - Analog 159
  - Acc 166
  - IRSound 176

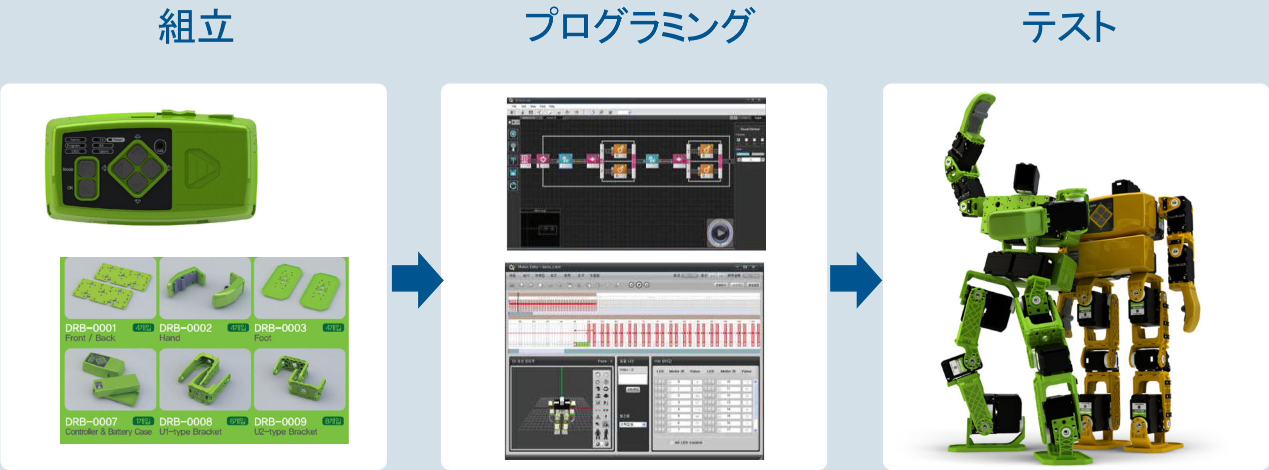
別添

- 有用な情報 191

PART 01

紹介

組立から動作検証テストまで


組立
プログラミング
テスト
DRB-0001 4개입 Front / Back
DRB-0002 4개입 Hand
DRB-0003 4개입 Foot
DRB-0007 1개입 Controller & Battery Case
DRB-0008 6개입 U1-type Bracket
DRB-0009 8개입 U2-type Bracket

■ 組立

ロボットを組み立てます。ロボットの組み立て方法は組立図で詳細に説明しています。DongbuRobotはモーター16DOF～20DOFの基本ヒューマノイドロボットの組立図を提供し、そのほかに26種類以上の拡張型ロボットを製作することができるキットで構成されています。様々なロボットの組立方法はウェブサイトでウェプマニュアルを提供しています。

■ プログラミング

DR-VisualLogicのソフトウェアを用いてロボットをプログラミングします。また、DR-SIMのソフトウェアを用いてロボットの動作を編集したり、キャプチャしてモーションを作成できます。作成したモーションはロボットにダウンロードしてすぐに実行することができます。また、DRCコントローラに保存し、DR- Visual Logicを使って読み込みプログラムすることができます。

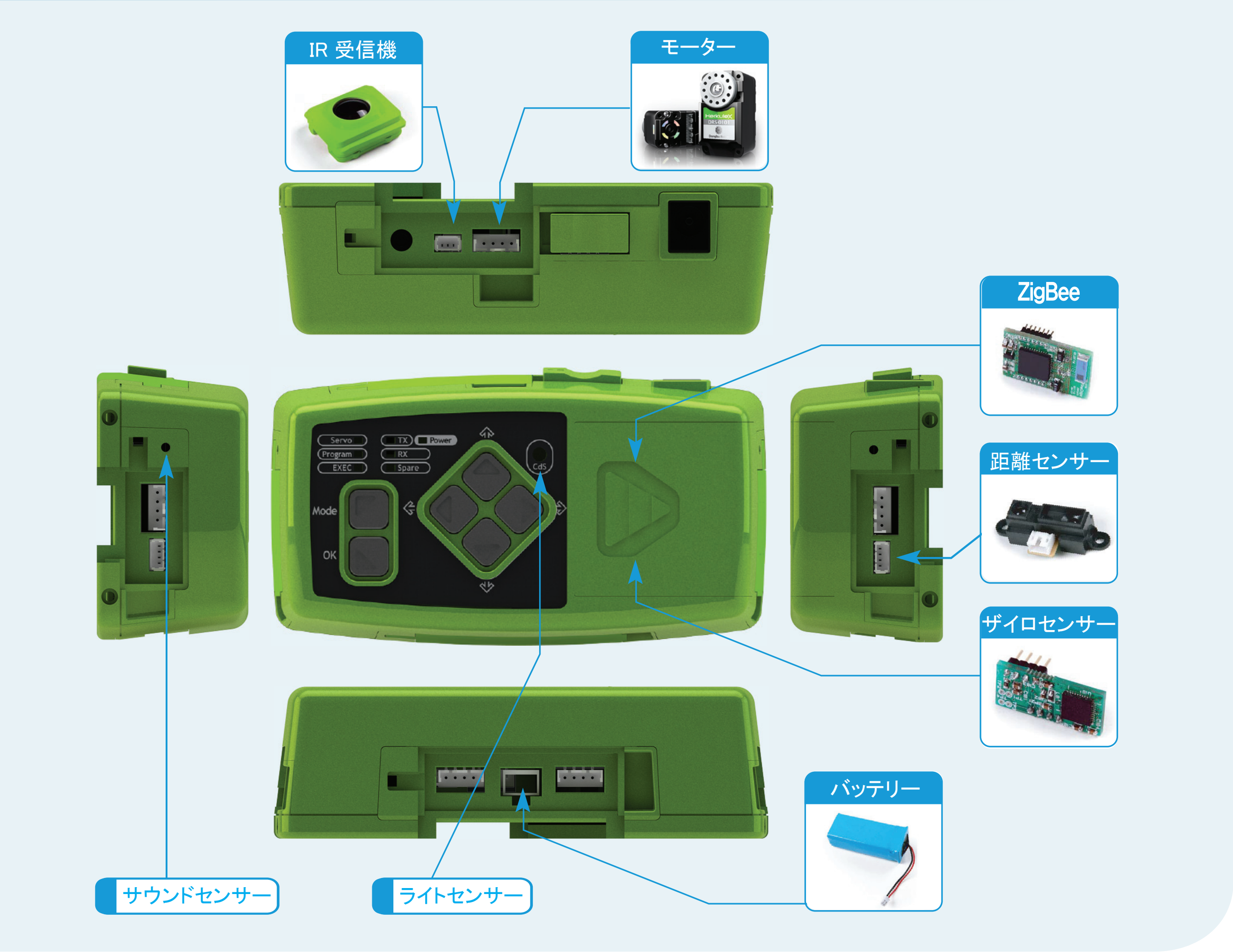
■ テスト

DR-Visual Logicで作成したプログラムをコンパイルしてロボットにダウンロードします。目的の動作を実行させることができなかった場合は再度モジュールのプロパティーを調整してからダウンロードして実行し、テストしてみてください。

コントローラー

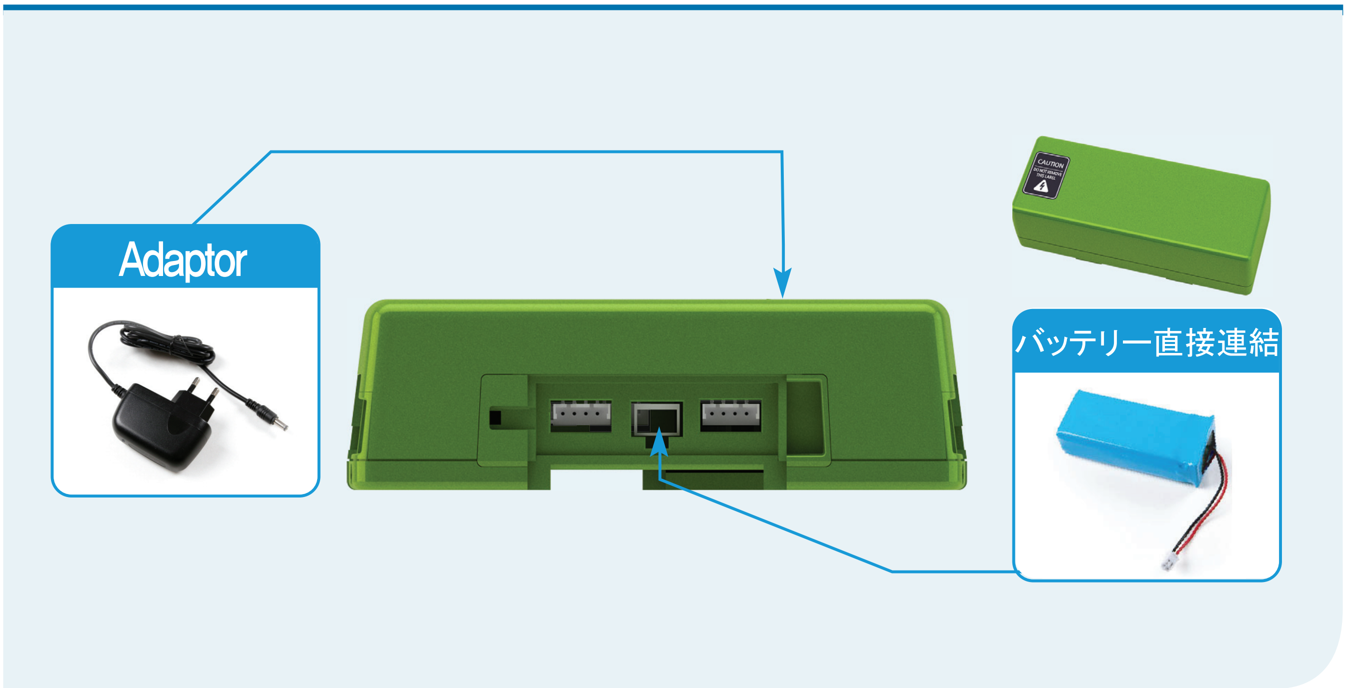
概要

コントローラーDRCはHovis　Liteの頭脳の役割をしているメイン部品です。コントローラーは6つのサーボモーターポート、2つのPSDセンサのポート、Gyroセンサ、ZigBeeを装着するためのコネクタなどのインタフェースで構成されています。また、光センサとサウンドセンサが組み込まれています。 DR-Visual　Logicプログラムは本DRCコントローラーの機能を利用してプログラミングできるビジュアルロボットのプログラミング言語です。各種センサのテストやモーターを1～32個まで接続してプログラミングできます。


IR 受信機
モーター
ZigBee
距離センサー
ザイロセンサー
バッテリー
サウンドセンサー
ライトセンサー
Servo
Program
EXEC
TX
Power
RX
Spare
Mode
OK
CdS

CPU	ATMega 128
Size, Weight	108 x 58.5 x 33 (mm), 82 g
Operating Voltage	Tolerance Range : 6.5V ～ 10V, Recommended Voltage : 7.4V
Serial Speed	115,200 bps ～ 666,667 bps
Consumed Current	When IDLE : 50mA, Overall Max Current : 3A (PTC Fuse)
Interface	Button : 6ea, MIC : 2ea, LED : 7ea
External I/O	Servo Motor : 6ea, PSD Sensor : 2ea
Back Cover I/O	ZigBee : 1ea, Gyro Sensor : 1ea
Internal I/O	Sound Sensor : 2ea, Light Sensor : 1ea

DRCバッテリーの取り付け方法

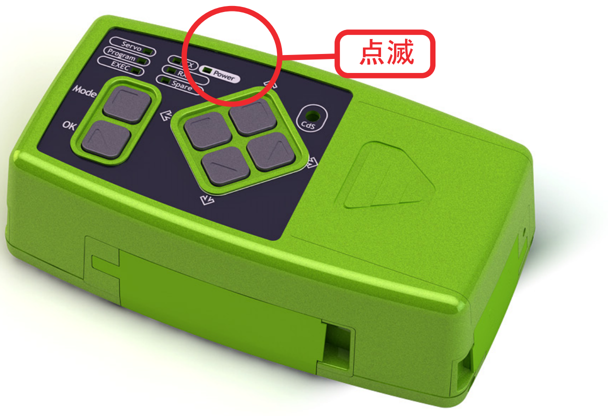

Adaptor
バッテリー直接連結

- バッテリー

電源をDRCに供給するために、電池とコントローラーの下段の電源ポートに接続します。

- 充電方法

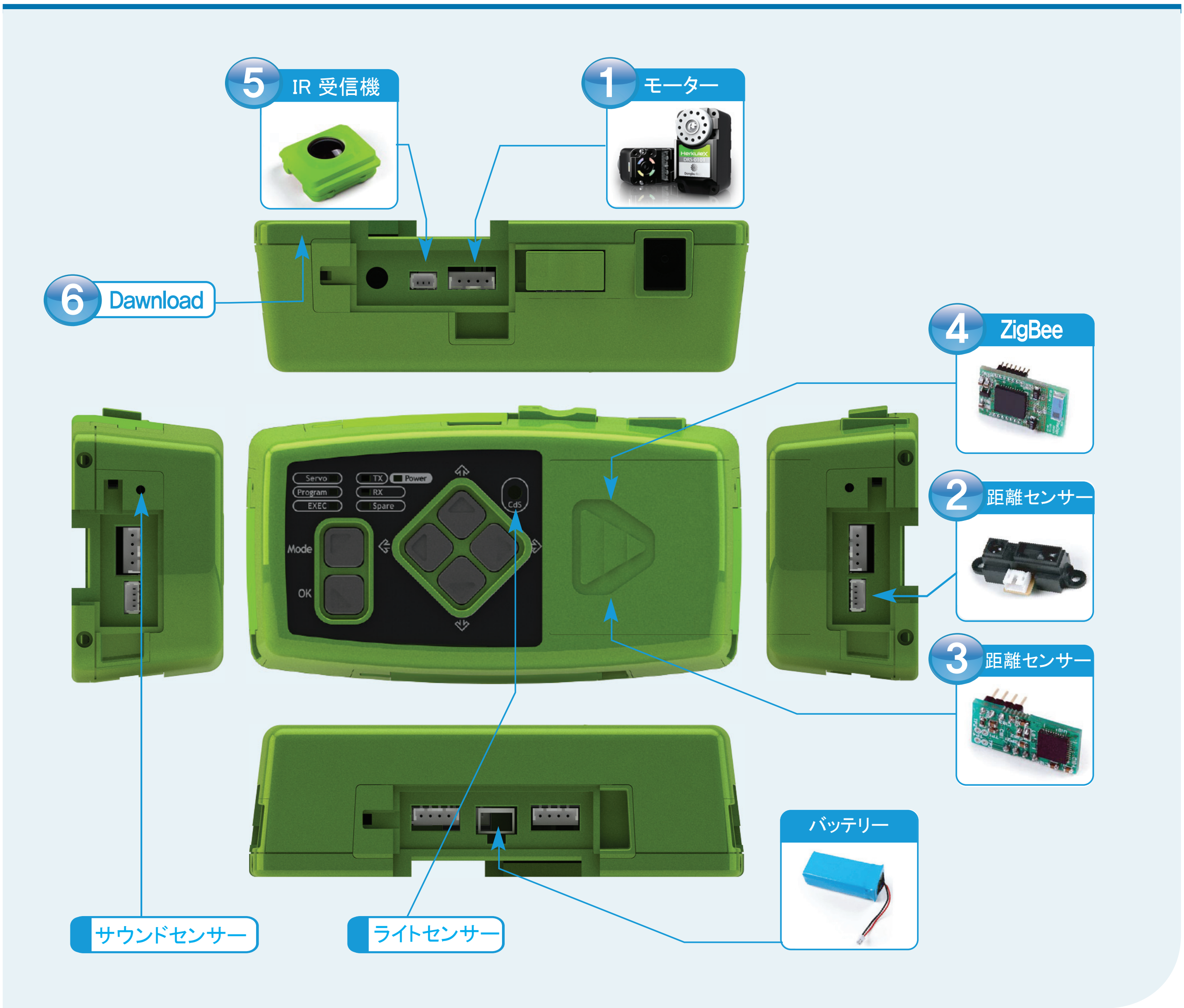
アダプターとバッテリーケーブルを利用して直接充電できます。
もう一つの方法はコントローラーの上段にあるアダプタのコネクタを利用することができます。そのコネクタにアダプタを差し込むと、コントローラーと接続されたバッテリが自動充電されます。


点滅
Servo
Program
EXEC
Mode
OK
Power
Spare
CdS

- バッテリー

バッテリー残量が20%以下の時、PowerLEDライトが点滅します。

DRCの接続方法


5 IR 受信機
1 モーター
6 Dawnload
Servo
Program
EXEC
TX
Power
RX
Spare
Mode
OK
CdS
4 ZigBee
2 距離センサー
3 距離センサー
バッテリー
サウンドセンサー
ライトセンサー

1 モーターの接続

コントローラーの周辺に全5つのポートがあります。

2 距離センサの接続

左、右側に2つのポートがあります。

3 ジャイロセンサの接続

コントローラーのスロットカバーを開き、内部に装着します。

4 ZigBeeの接続

コントローラーのスロットカバーを開き、内部に装着します。

5 IR Receiverの接続

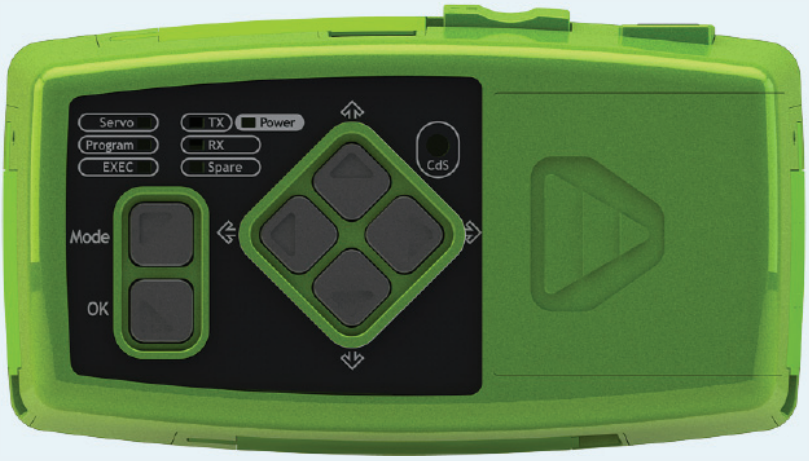
1つのポートがあります。リモコン受信用です。

6 ダウンロードリンク

Ear phone Jack でPCプログラムをロボットに
ダウンロードする際に使います。

コントローラー

DRCのインタフェース



DRCコントローラーはコントローラー正面にInput/Outputを示すボタンとLEDがあり、側面はモーターやセンサを接続するポートで構成されています。DRCのUserInterfaceは正面のボタンの操作でコントローラーにInput命令をすることができ、LEDでOutputのデータを表示できます。

	表記	ショートカットキー	基本Taskモード
Main Button	Mode	Task実行	
	Ok	確認ボタン	
Navi Key	（左）	バッテリーの残量表示	点検モード
	（上）	DRC自己診断	自律行動モード
	（右）	有線/無線の切り替え	リモコン操縦モード
	（下）	モーターのIDスキャン	サウンドデモモード
LED	Servo	HerkuleX 実行中	
	Program	DR-SIM/Visual Logic 実行中	
	EXEC	Task 実行中	
	TX	データ送信	
	RX	データ受信	
	Spare	ユーザーの割り当て	
	Power	電源表示	
Sensor	Cds	Light Sensor	

DRC Register Map & Protocol

■ レジスタ

DRCにはレジスタと呼ばれる領域があります。コントローラーの現在の状態、動作設定、センサに関連したいくつかのデータが入っています。例えば現在のコントローラーに接続しているサーボモーターの数、モーターのID、コントローラーのエラー状態、エラーコードなどはコントローラーの現在の状態に該当します。コントローラーが動作可能な最小/最大電圧、命令の応答設定はコントローラーの動作設定に該当します。また、光センサの現在の光量、サウンドセンサが感知した音の方向性などはセンサのデータに該当します。

コントローラーのレジスタは不揮発性(Non-Volatile、EEPROM)Registerと揮発性(Volatile、RAM)Registerに区別されます。不揮発性レジスタは電源がオフの状態でもデータは消去されない領域で、コントローラーの動作の基本設定値が保存されています。不揮発性レジスタの値は電源が入った瞬間に揮発性レジスタの対応する領域にコピーされます。揮発性レジスタはコントローラーの動作設定、コントローラーの状態、センサ値を保存しており、揮発性レジスタに保存した動作設定がコントローラーの動作に直接影響を与えます。

レジスタの各項目がどのように変化するかを知っていればDR-Visual Logicでロボットの行動をプログラムする時、さらに細かな動作条件を作成することができます。また、コントローラの状態を読み取ったり、設定を変えることでロボットを容易に使えるようになります。詳細についてはDRCレジスタとプロトコルの詳細なマニュアルを参照してください。

■ プロトコル

ユーザーがレジスタの値を読み取ったり、変更する場合はそれを指示する命令を送信する必要があります。その命令の決めているフォーマットをプロトコルと呼びます。プロトコルにはレジスタを読み書きする命令、コントローラーに保存されたモーションタスク、メロディなどを実行するコマンド、コントローラーを再起動するコマンドなど様々な命令が定義されています。

PCとコントローラーの通信時にはこのように定義されたプロトコルを通して要求パケットと応答パケットを送受信します。Dongbu Robotが提供するDR-SIMとDR-Visual Logicもこのプロトコルを利用して作られています。ユーザーはプロトコルを熟知することで、自分で作成したプログラムでDRCを制御することができます。

詳細についてはDRCレジスタとプロトコルの詳細なマニュアルを参照してください。

DRC機能の使い方

1 プログラムの定義

Firmware：ユーザーが変更できない内部プログラムです。

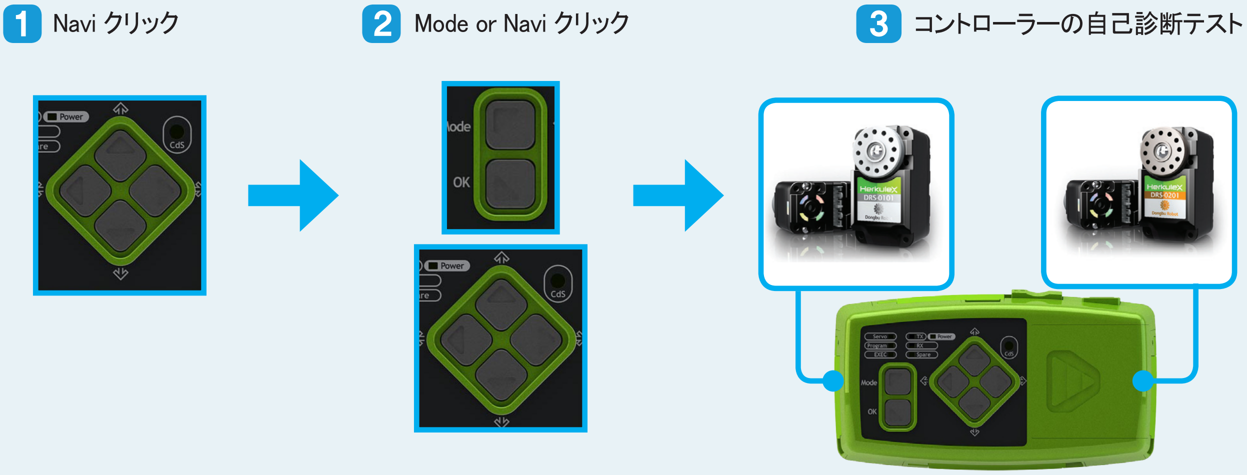
Task：Task Editor（DR-Visual Logic）を使用してユーザーが変更できるユーザー定義のプログラムです。

出荷時にはヒューマノイド専用の基本Task Programが定義されており、ユーザーが変更可能です。

2 動作方法

電源を入れNavi Keyを押すと、Firmwareに入っている基本機能が実行され、Modeボタンを押すと、保存されているTaskが実行されます。基本Taskの場合、Task実行後、再びNavi Key Okをクリックして、どのモードに入るか選択することができます。

Firmware（ユーザーが変更できません）

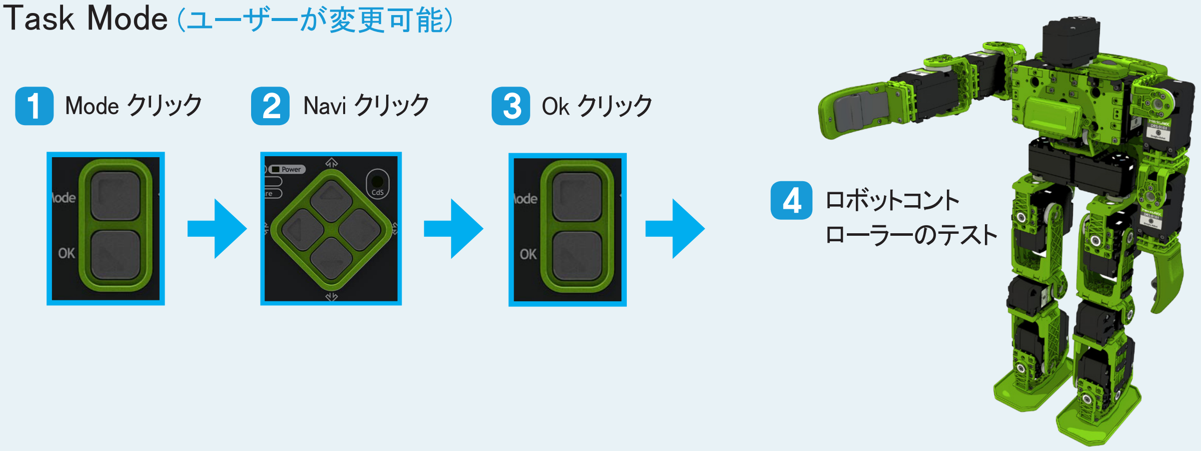


1 Navi クリック

2 Mode or Navi クリック

3 コントローラーの自己診断テスト

Task Mode（ユーザーが変更可能）



1 Mode クリック

2 Navi クリック

3 Ok クリック

4 ロボットコントローラーのテスト

3 機能の使い方や動作

	表記	説明	機能や動作方法
Main Button	◩ Mode	モード変更	Taskを始める 基本Taskの場合、Mode → Navi key → Okで動作モードを選択可能
	◣ OK	確認ボタン	
Navi Key (Firmware)	◀ (左)	バッテリーの残量表示	バッテリーの残量確認が可能→　左LEDの3つの表示で最も低い時に1つ、中間の時に2つ、最も高い時に3つ
	▲ (上)	テスト	コントローラーでモーターやセンサのテスト 使い方 : (上)→ ボタン → センサによるモーターの反応でテスト テスト項目 : モーター/光/音/距離/加速度/ジャイロセンサ
	▶ (右)	有線/無線の切り替え	Ear phone jackを経由して有線通信/ ZigBeeによる無線通信の切り替え
	▼ (下)	モーターのIDスキャン	現在接続しているモーターのIDを再スキャン
Navi Key (ヒューマノイド専用の基本Task)	◀ (左)	点検モード	Mode → (左) → Ok : 点検モード : モーターを一つずつ選択し、そのIDのモーターが存在しない場合はビープ音を出す
	▲ (上)	自律行動モード	Mode → (上) → Ok : 自律行動モード : ロボットが動きながら、自ら動作
	▶ (右)	リモコン操縦モード	Mode → (右) → Ok : リモコン操縦モード : リモコンで定義されている動作を実行
	▼ (下)	サウンドデモモード	Mode → (下) → Ok : サウンドデモモード : サウンド入力に応じて様々な動作をする
LED_mode	Servo	HerkuleX 実行中	HerkuleX Managerが実行中の時に点滅
	Program	DR-SIM/Visual Logic 実行中	DR-SIM/ Visual Logicを使用して、編集、使用中の時に点滅する。 データやファームウェアをダウンロードする時は、オンになる
	EXEC	モード変更/Task 実行中	Modeボタンを押して、Taskモードに入る時、Taskの実行中の場合オン
	TX	データ送信	外部へデータを送信する時、点滅する、task実行中の時はUser Spare空間
	RX	データ受信	外部からのデータを受信する時、点滅する、task実行中の時はUser Spare空間
	Spare	ユーザーの割り当て	
	LED点滅	エラー発生	実行中、エラーが生じた場合は右の3つのLEDが点滅しながら通知
LED_Power	Power	電源表示	残量が20%未満の時、点滅
Sensor	CdS	Light Sensor	光感知センサ

DRC基本テスト

DRCはロボットを組み立てなくてもテストモードを使用して基本テストが可能です。電源を入れ、コントローラーのショートカットキー(上)ボタンを押してモーターとセンサのテストを実行します。センサテストはモーターID1と2をそれぞれ左、右に挿入した状態でモーターの反応でテストを実行します。テスト項目はモーター、光センサ、音センサ、距離センサ、ジャイロセンサなどがあります。シナリオと操作方法の概要は以下の通りです。

光センサ、音センサが内蔵されている機能なので、menuとokでテストします。
PSD、Acc/ Gyroは（下）キーを押すと、まず、各センサが接続されているか一つずつチェックして、一つだけ接続されている場合は処理し、二つが同時に接続されている場合は処理されません。

1 モーター :左、右に一つずつモーターを差し込む（左に1番、右に2番）
（上）➔（左）:左側のモーターが繰り返し動く。（右）:右のモーターが動く。（上）:モーターの二つが同時に動く。

2 光センサ :（上）➔（Menu）:Light機能が起動、Cdsウィンドウを遮ると、モーターの二つが同時に動く。

3 音センサ :（上）➔（OK）: Sound Sensor起動
左側で拍手すると、左側のモーターが動き、
右側で拍手すると、右側のモーターが動く。

以下の機能はセンサを接続後（下）ボタンを押すと動作します。テストするセンサを1台だけ接続します。2つ以上が接続されている場合は動作しません。

4 PSD Digital :（上）➔ デジタル距離センサの接続 ➔（下）PSD起動
10cm以下の物体が接近すると ➔ モーター二つが動き
10cm以上の物体が遠くなれば（崖の検出）➔モーター二つが停止
→ 崖の検出

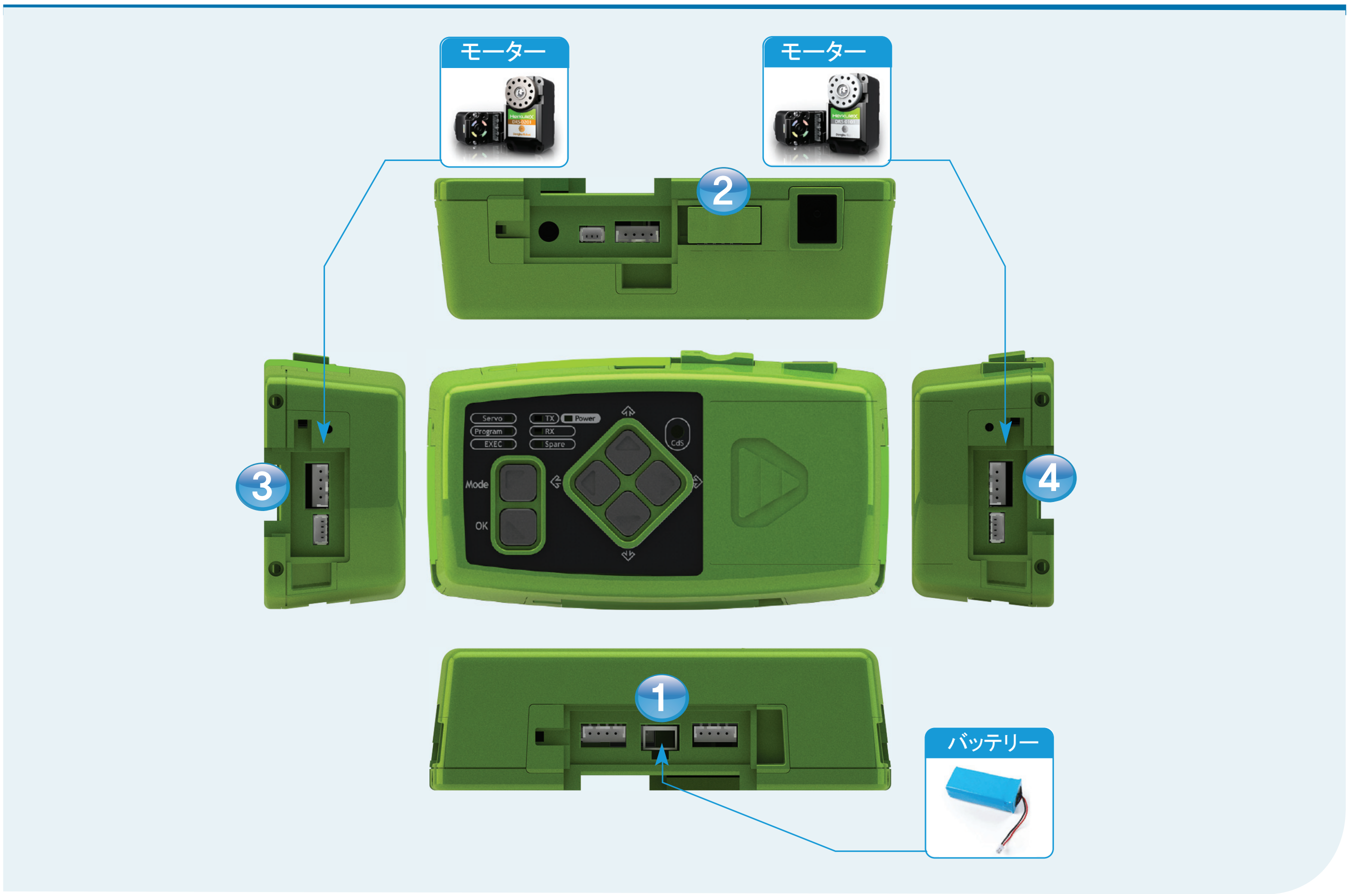
5 PSD Analog :（上）➔アナログPSDセンサの接続 ➔（下）PSD起動
作動すると同時に二つのモーターが同じ方向に動く。物体との距離が遠いほど、より速く動いて、
物体が近ければゆっくり動く ➔ 物体が10 cm以内に接近するとモーターが逆に動き
→ 壁を検出し、避けるシナリオ

6 Acc :（上）➔ Acc/Gyro装着 ➔（下）Acc起動
コントローラーのAccセンサがまっすぐに立った状態ではモーターが止まっている。コントローラーの傾きに応じてモーターは速度を変えて動く。コントローラーをさらに傾けると速く動く。

7 Gyro :（上）➔ Acc/Gyro 装着 ➔（下）Acc 起動 ➔（下）Gyro 起動
コントローラーが停止している時には動かない。
コントローラーが回る速度に合わせてモーターも一緒に動く。

次の詳細なテストページを参照して確実に実行してください。

DRC基本テスト：モーター


モーター
モーター
2
3
Servo
Program
EXEC
TX
Power
RX
Spare
CdS
Mode
OK
4
1
バッテリー

1 バッテリーの接続

2 電源を入れる

3 **左モータの接続**：必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 **右モータの接続**：必ずMotor ID2番を接続してください（他のモーターは動作しません。）

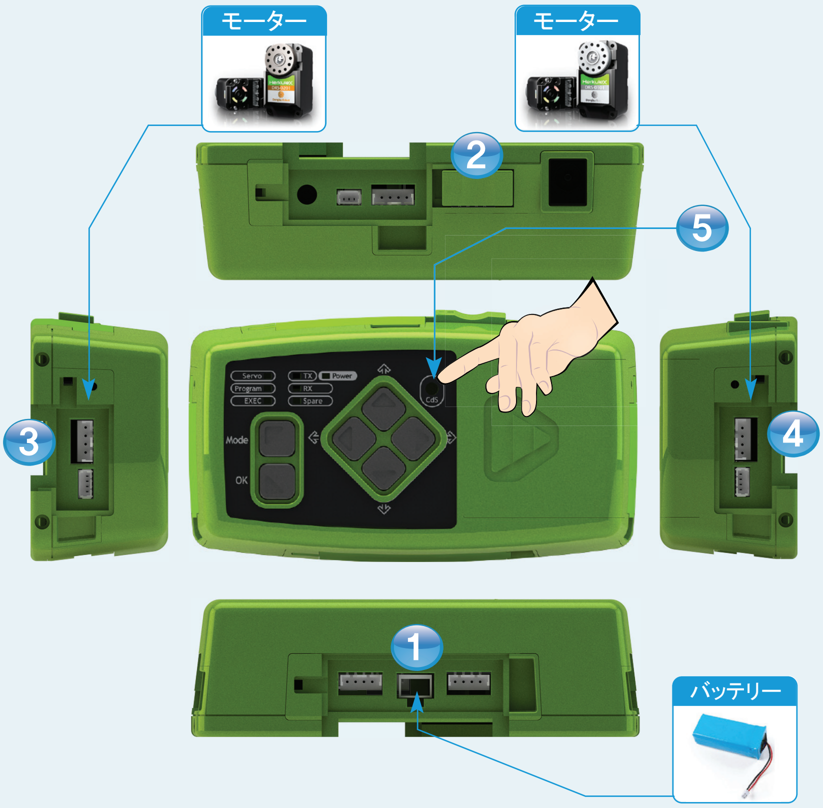
モーターは外に置いてテストします。車輪が回ることを想定したシナリオです。

→ テストの進行

- 電源を入れて（上）ボタンを押して、Test Mode切換後
- Navi Keyの（左）ボタンをクリックします。→ 左側のモーターが動作します。
- Navi Keyの（右）ボタンをクリックします。→ 右側のモーターが動作します。
- Navi Keyの（上）ボタンをクリックします。→ 二つのモーターが前進方向に動作します。

上と同じように動作した場合、モーターが異常なく動作したことを意味します。モーターテストを完了します。以降のすべてのテストの結果は上の二つのモーターが正常に動作するかどうかで判断します。モーターを分離せず、センサテストを続行します。

DRC基本テスト：光センサ


モーター
モーター
2
5
Servo
Program
EXEC
TX
Power
RX
Spare
Mode
OK
CdS
3
4
1
バッテリー

1 バッテリーの接続

2 電源を入れる

3 **左モータの接続**：必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 **右モータの接続**：必ずMotor ID2番を接続してください（他のモーターは動作しません。）

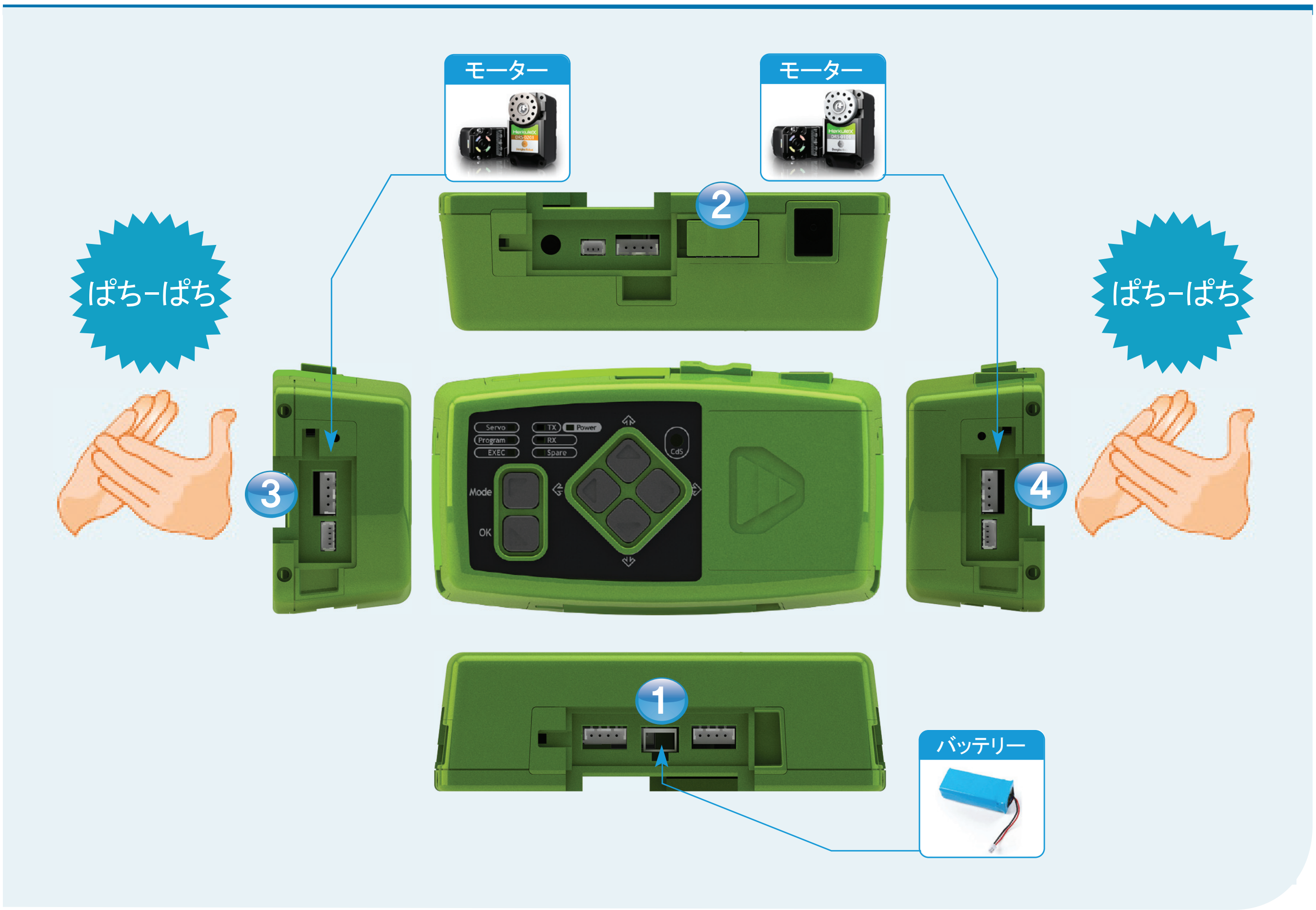
光が消えた時、ロボットが腕で虚空を掴むの仮想シナリオです。

→ テストの進行

- 電源を入れて（上）ボタンを押して、Test Mode切換後
- Cdsウィンドウを手のひらで覆います。➔左右のモーターが同時に動きます。

上と同じように動作した場合、光センサが異常なく動作したことを意味します。光センサのテストを完了します。

DRC基本テスト：音センサ


モーター
モーター
2
ぱち-ぱち
ぱち-ぱち
3
4
Servo
Program
EXEC
TX
Power
RX
Spare
Mode
OK
CdS
1
バッテリー

1 バッテリーの接続

2 電源を入れる

3 **左モータの接続**：必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 **右モータの接続**：必ずMotor ID2番を接続してください（他のモーターは動作しません。）

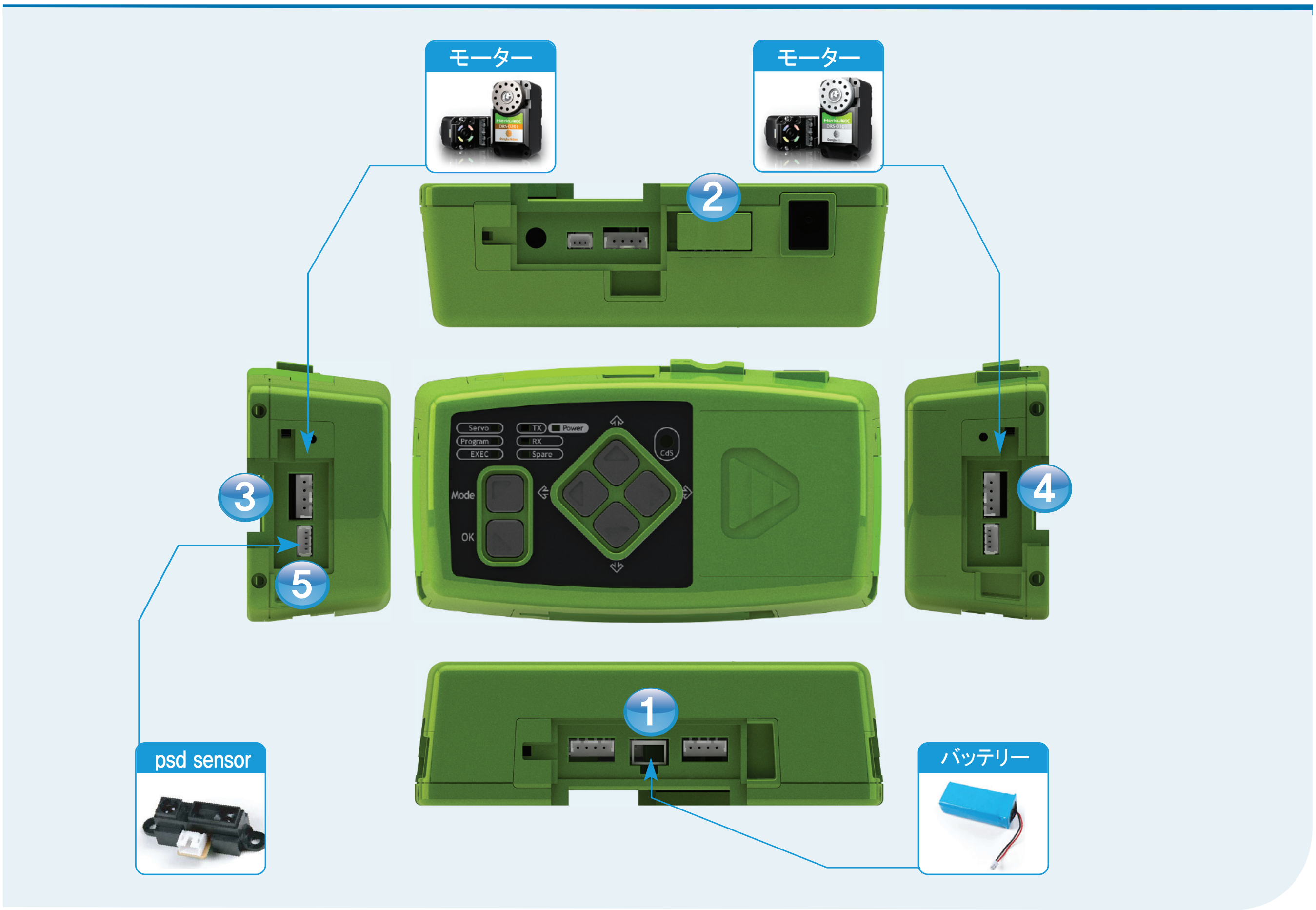
音センサはコントローラーの左右に設置されています。拍手する方向のモーターが動くシナリオです。

→ テストの進行

- 電源を入れて（上）ボタンを押して、Test Modeに切換後。
- （OK）ボタンをクリックします。→ 音センサが起動します。
- 左側で拍手をする。→ 左側のモーターが動きます。
- 右側で拍手をする。→ 右側のモーターが動きます。

上と同じように動作した場合、音センサが異常なく動作したことを意味します。音センサのテストを完了します。

DRC基本テスト：PSD デジタル距離センサ



1 バッテリーの接続

2 電源を入れる

3 左モータの接続：必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 右モータの接続：必ずMotor ID2番を接続してください（他のモーターは動作しません。）

5 PSDデジタルセンサの接続

PSDデジタルセンサは一定の距離を基準に距離を測定するセンサです。
一般的に地面の深さをチェックして崖を感知した時にロボットを停止させるシナリオです。

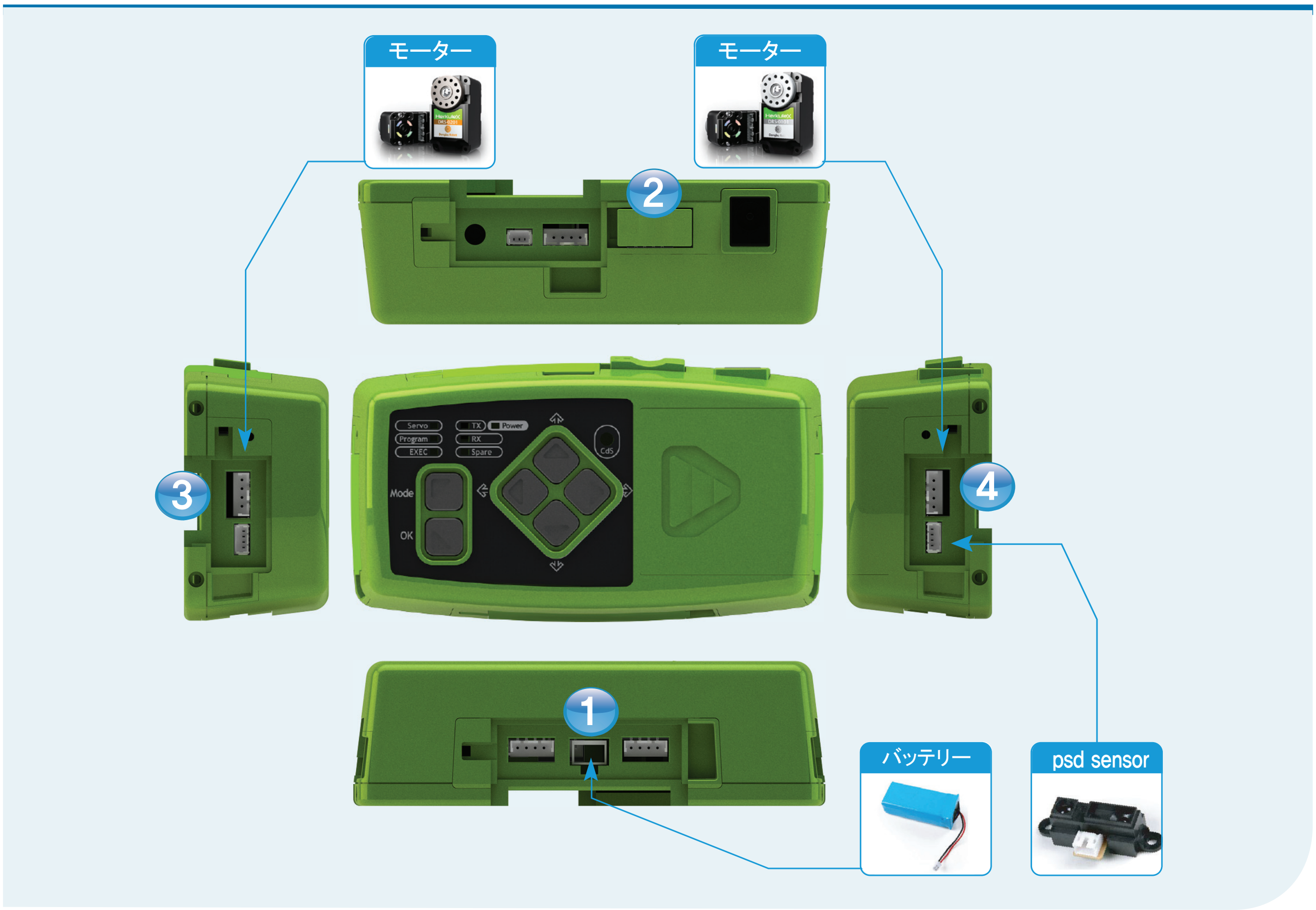
→ テストの進行

- 電源を入れて（上）ボタンを押して、Test Modeに切換後
- PSD線を接続して（下）ボタンをクリックします。→ PSDデジタルセンサが起動します。
- 手のひらや物体がセンサから10cm以下の場合→ 両方のモーターが動きます。
- 手のひらや物体がセンサから10cm以上の場合→ 両方のモーターが停止します。

＊PSDのデジタルセンサは基準の位置に基づいて、On/ Off形式でのみ動作します。

上と同じように動作した場合、PSDデジタルセンサが異常なく動作したことを意味します。PSDデジタルセンサのテストを完了します。

DRC基本テスト：PSD アナログ距離センサ


モーター
モーター
2
Servo
Program
EXEC
TX
Power
RX
Spare
CdS
Mode
OK
3
4
1
バッテリー
psd sensor

1 バッテリーの接続

2 電源を入れる

3 左モータの接続：必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 右モータの接続：必ずMotor ID2番を接続してください（他のモーターは動作しません。）

5 PSDのアナログセンサの接続

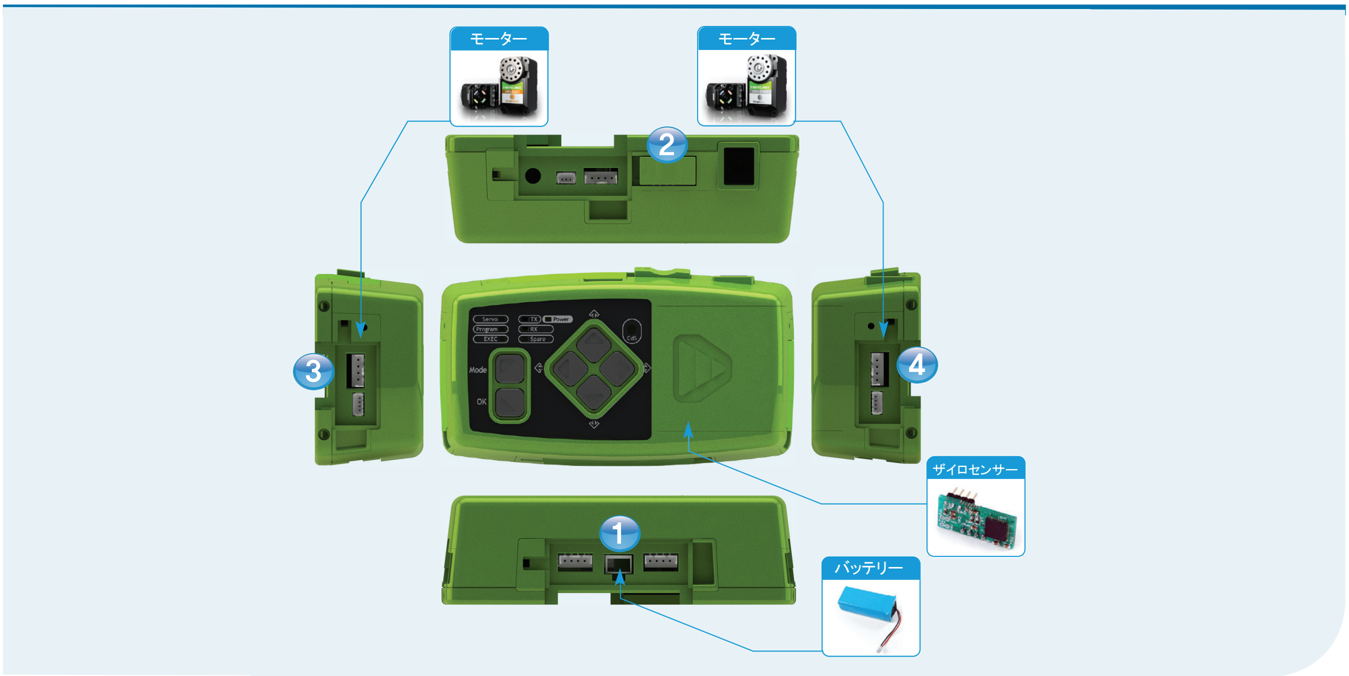
PSDのアナログセンサはリアルタイムで距離測定が可能です。接近距離別にモーターの速度を調整します。ほとんどの障害物の検出に使われ、障害物が近くに来た時に速度を落として、回避する動作に使用されます。

→ テストの進行

- 電源を入れて（上）ボタンを押して、Test Mode切換後、PSD線を接続して、（下）ボタンをクリックします。
  → PSDデジタルセンサが起動します。 → 二つのモーターが同じ方向に移動します。
- 手のひらや物体をPSDセンサの前に配置し、その後、遠くに移動させます。 → 物体が遠いほど、モーターは速く動きます。
  → 物体が近ければ近いほどモーターはゆっくり動きます。
- 物体が5cm以内に接近すると、モーターは逆に動きます。

上と同じように動作した場合、PSDアナログセンサが異常なく動作したことを意味します。PSDアナログセンサのテストを完了します。

DRC基本テスト : 加速度/ジャイロセンサ



1 バッテリーの接続

2 電源を入れる

3 **左モータの接続** : 必ずMotor ID1番を接続してください（他のモーターは動作しません。）

4 **右モータの接続** : 必ずMotor ID2番を接続してください（他のモーターは動作しません。）

モーターは外に置いてテストします。車輪が回ることを想定したシナリオです。

5 **Acc/ Gyroセンサの接続**: ロボットコントローラーのふたを開け、加速度/ジャイロセンサを接続します。

→ テストの進行 : ACC

- 電源を入れて（上）ボタンを押して、Test Mode切換後、Acc/Gyroを装着し、（下）ボタンを押します。
  → 加速度センサが起動します。
- コントローラーに搭載したAccセンサがまっすぐに立った状態であれば、モーターが停止します。
- コントローラーを徐々に傾けます → 傾きに応じた速度でモーターが動きます。さらに傾けると、速度が速くなります。

上と同じように動作した場合、Accセンサが異常なく動作したことを意味します。

→ テストの進行 : Gyro

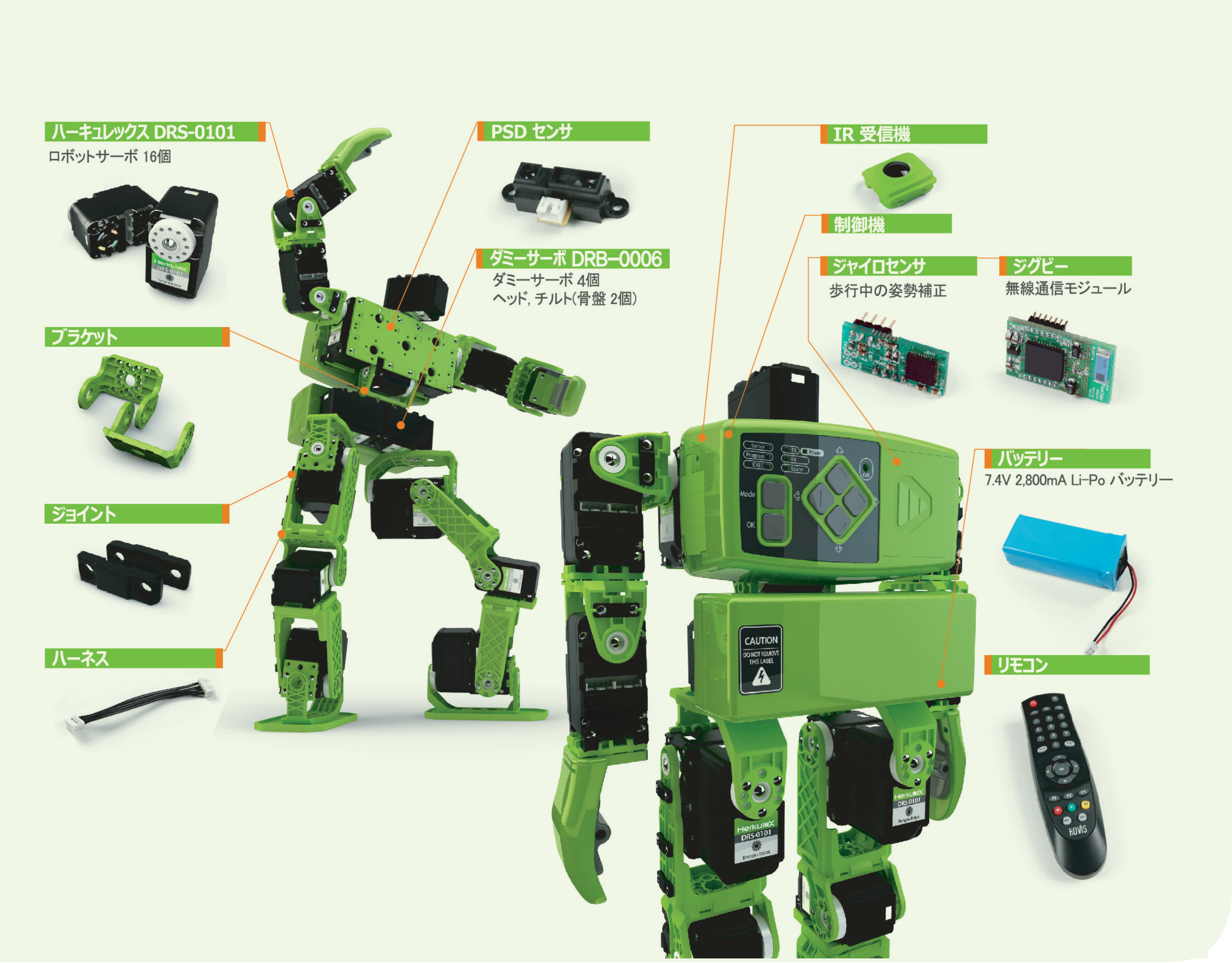
- ACCのテストモードの状態でもう一回（下）ボタンを押します。→ ジャイロセンサが起動します。
- コントローラーが止まっている時はモーターも停止します。
- コントローラーを動かします。→ コントローラーが回る速度に合わせてモーターの速度が変わります。

上と同じように動作した場合、加速度/ジャイロセンサが異常なく動作したことを意味します。 加速度/ジャイロセンサのテストを完了します。AccとGyroは1つのチップにモジュール化されています。

PART
03

部品リスト

Hovis Lite 構成図


ハーキュレックス DRS-0101
ロボットサーボ 16個
ブラケット
ジョイント
ハーネス
PSD センサ
ダミーサーボ DRB–0006
ダミーサーボ 4個
ヘッド, チルト(骨盤 2個)
IR 受信機
制御機
ジャイロセンサ
歩行中の姿勢補正
ジグビー
無線通信モジュール
バッテリー
7.4V 2,800mA Li–Po バッテリー
リモコン

1. サーボモーター:ハーキュレックスDRS–0101
2. ブラケット:サーボモーター間の接続、関節の役割
3. ジョイント:
4. ハーネス:ケーブル
5. PSDセンサ:距離測定
6. IR受信機:リモコン受信機
7. DRCコントローラー:
8. Gyroセンサ:歩行中の姿勢修正
9. ZigBee:通信モジュール
10. バッテリー:
11. リモコン:

部品リスト

Hovis Lite 部品リスト

部品別数量です。

Main 部品	Controller	1個入	□
	7.4V Li–po Battery	1個入	□
	AC adoptor	1個入	□
	HerkuleX DRS–0101	16個入	□
ブラケット	Front / Back	4個入	□
	Hand	4個入	□
	Foot	4個入	□
	Universal Plate	40個入	□
	Ankle Plate	8個入	□
	Dummy Servot	4個入	□
	U1–type Bracket	6個入	□
	U2–type Bracket	8個入	□
	U3–type Bracket	6個入	□
	U4–type Bracket	8個入	□
アクセサリ	Serial Cable (DSUB 9Pin/3p Audio Jack)	1個入	□
	USB to Serial Gender	1個入	□
	Wheel (White, Ø60)	4個入	□
	Horn(Plastic)	5個入	□

ジョイント	L–type Joint (Single Nut)	70個入	□
	L–type Joint (Double Nut)	10個入	□
	L–type Joint (Hole Only)	40個入	□
	I–type Joint (10.9mm,Hole only)	10個入	□
	I–type Joint (12.5mm,Double Nut)	60個入	□
	I–type Joint (16.0mm,Hole only)	20個入	□
	V–type Joint (12.0mm,Single Nut)	20個入	□
	V–type Joint (12.0mm,Double Nut)	20個入	□
	Bushing Set	30個入	□
ボルト & ナット & ハーネス	Nut (M2)	40個入	□
	Nut (M3)	10個入	□
	Bolt (PH/T 2.0X13)	25個入	□
	Bolt (PH/M 2.0X4)	50個入	□
	Bolt (PH/M 2.0X5)	20個入	□
	Bolt (PH/M 2.0X6)	150個入	□
	Bolt (PH/M 2.0X8)	20個入	□
	Bolt (PH/M 3.0X6)	10個入	□
	Bolt (PH/M 3.0X8)	20個入	□
	Bolt (PH/T 2.0X4)	40個入	□
	Bolt (PH/T 2.0X5)	60個入	□
	Harness (75mm)	4個入	□
	Harness (100mm)	4個入	□
	Harness (200mm)	6個入	□
	Harness (300mm)	2個入	□
	Harness Clamp	20個入	□

部品リスト

Hovis Lite 部品リスト

Main 部品

DRC–005T 1個入
Controller

DRL–0728 1個入
7.4V Li–Po Battery
(3,000mA)

DRQ–0002 1個入
AC Adaptor

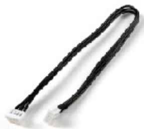
DRS–0101 16個入
HerkuleX Smart Servo

ボルト & ナット & ハーネス

DRA–0002 4個入
Harness (75mm)

DRA–0003 4個入
Harness (100mm)

DRA–0006 6個入
Harness (200mm)



DRA–0007 2個入
Harness (300mm)

DRA–0051 40個入
Nut (M2)

DRA–0052 10個入
Nut (M3)

DRA–0054 25個入
Bolt (PH/T 2.0X13)

DRA–0056 50個入
Bolt (PH/M 2.0X4)

DRA–0057 20個入
Bolt (PH/M 2.0X5)

DRA–0058 150個入
Bolt (PH/M 2.0X6)

DRA–0059 20個入
Bolt (PH/M 2.0X8)

DRA–0061 10個入
Bolt (PH/M 3.0X6)

DRA–0062 20個入
Bolt (PH/M 3.0X8)

DRA–0063 40個入
Bolt (PH/T 2.0X4)

DRA–0064 60個入
Bolt (PH/T 2.0X5)

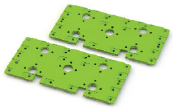
DRJ–0010 20個入
Harness Clamp

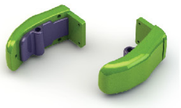
部品リスト

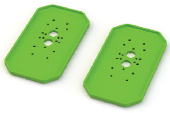
Hovis Lite 部品リスト

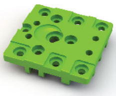
部品別写真です。

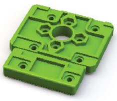
ブラケット

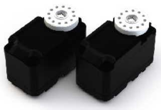

DRB-0001 4個入
Front / Back

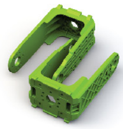

DRB-0002 4個入
Hand

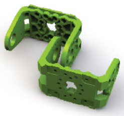

DRB-0003 4個入
Foot

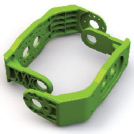

DRB-0004 40個入
Universal Plate

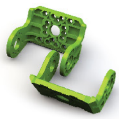

DRB-0005 8個入
Ankle Plate


DRB-0006 4個入
Dummy Servo

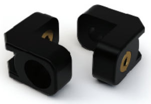

DRB-0008 6個入
U1-type Bracket

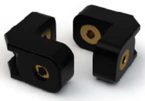

DRB-0009 8個入
U2-type Bracket

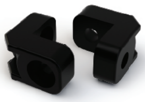

DRB-0010 6個入
U3-type Bracket

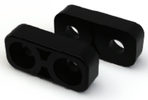

DRB-0011 8個入
U4-type Bracket

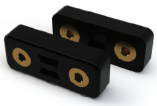
ジョイント

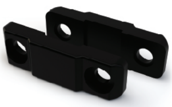

DRJ-0001 70個入
L-type Joint (Single Nut)

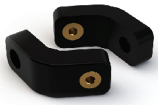

DRJ-0002 10個入
L-type Joint (Double Nut)

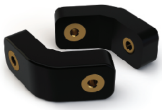

DRJ-0003 40個入
L-type Joint (Hole only)

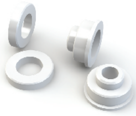

DRJ-0004 10個入
I-type Joint
(10.9mm, Hole only)


DRJ-0006 60個入
I-type Joint
(12.5mm, Double Nut)

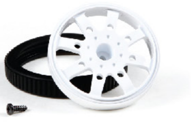

DRJ-0007 20個入
I-type Joint
(16.0mm, Hole only)

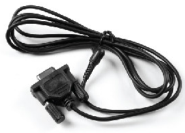

DRJ-0008 20個入
V-type Joint
(12.0mm, Single Nut)

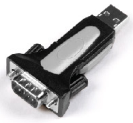

DRJ-0009 20個入
V-type Joint
(12.0mm, Double Nut)

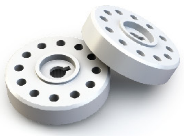

DRJ-0011 30個入
Bushing Set

アクセサリー


DCW-0001 4個入
Wheel (White, Ø60)


DRI-0002 1個入
Serial Cable


DRI-0003 1個入
USB to Serial Gender


DRH-1001 5個入
Horn (Plastic)

Humanoid 組立図

ヒューマノイドIDの配置と配線図です。

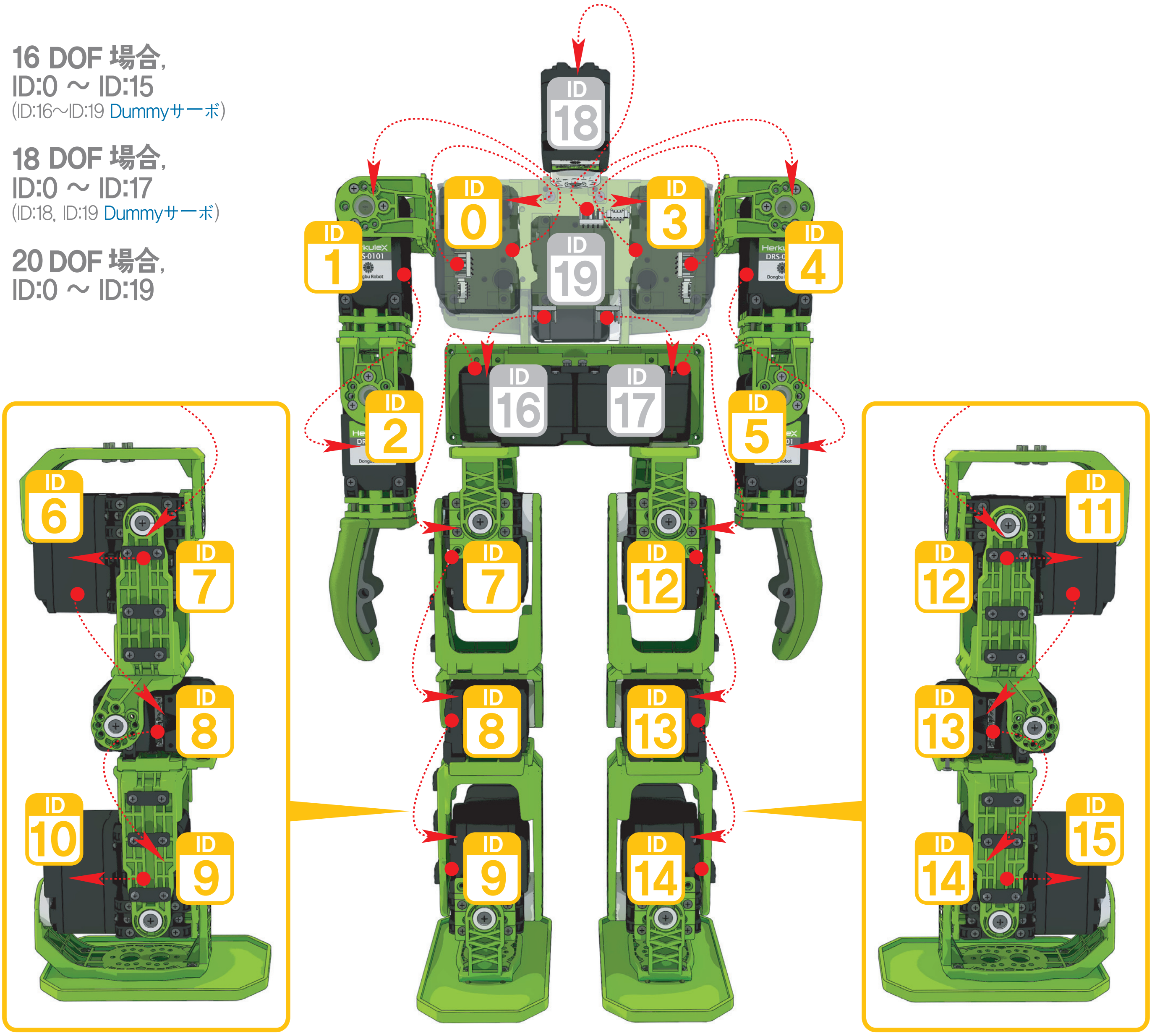
ヒューマノイドIDは製品出荷時にそれぞれのサーボモーターにIDが付与されて提供されます。モーターの外側に書かれたID番号の通りに組み立てることでロボットを正常に動作させることができます。 IDの配置基準は20DOFが基準です。16～19番はダミーモーターとの交換用なので、最後の番号に配置されています。ヒューマノイドの組み立て時に最も難易度の高いパーツがサーボ間の配線です。接続の全体像を考えた後に詳細に一つずつ試してください。配線の詳細な接続はヒューマノイドの組み立て図を参照してください。

---

16 DOF 場合,
ID:0 ～ ID:15
(ID:16～ID:19 Dummyサーボ)

18 DOF 場合,
ID:0 ～ ID:17
(ID:18, ID:19 Dummyサーボ)

20 DOF 場合,
ID:0 ～ ID:19

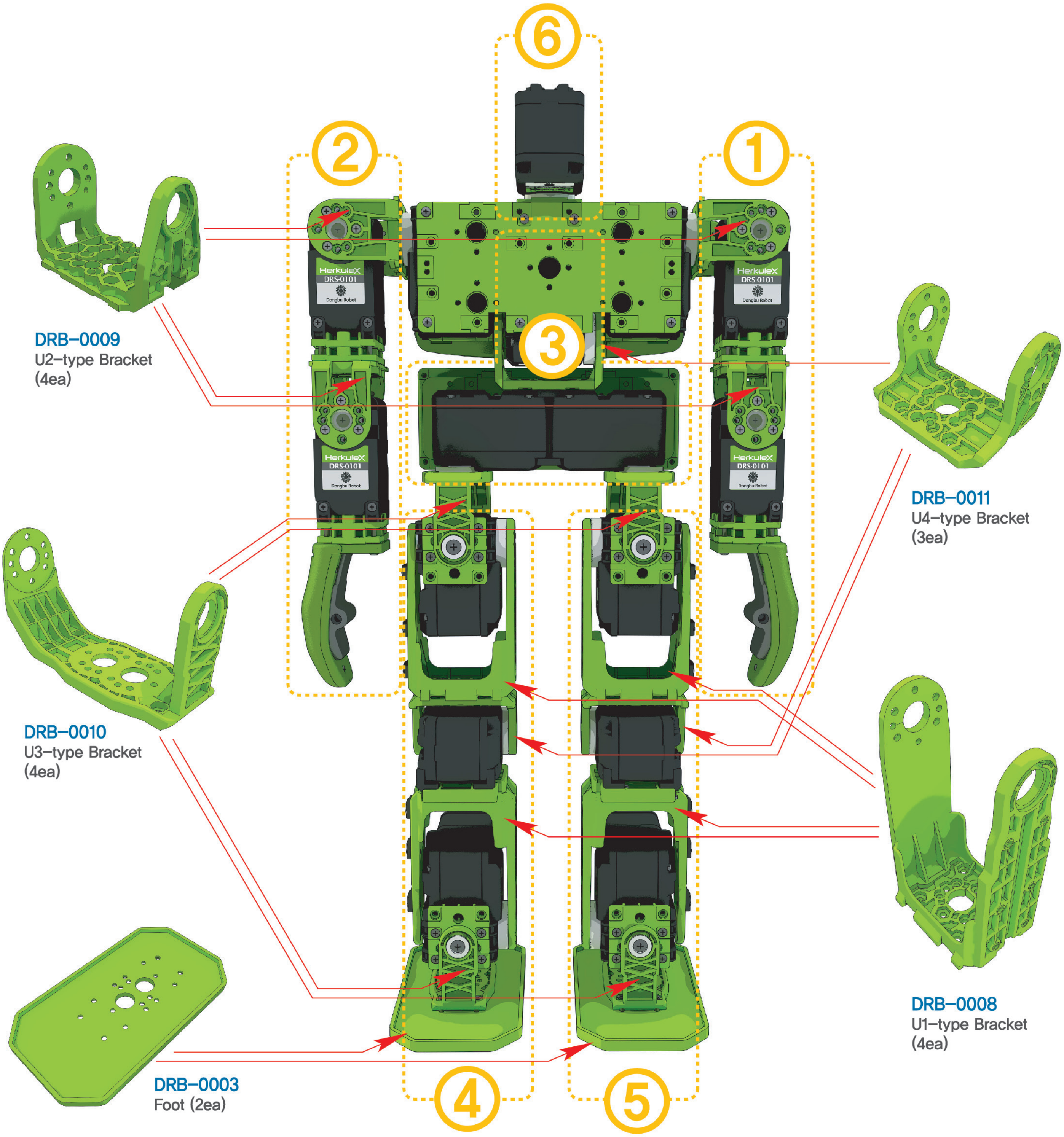

ID 18
ID 0
ID 3
ID 1
ID 19
ID 4
ID 16
ID 17
ID 2
ID 5
ID 6
ID 7
ID 8
ID 10
ID 9
ID 7
ID 12
ID 8
ID 13
ID 9
ID 14
ID 11
ID 12
ID 13
ID 15
ID 14

組立図

Humanoid 組立図

ヒューマノイドのブラケット配置図です。

ブラケットはサーボ間の接続用として用いられており、ロボットの関節として機能します。肩、腰、尻、膝、足の裏で構成され、全体のブラケット配置図を考えた後、詳細な組み立てに入ります。詳細な組み立て方法はヒューマノイドの組立図を参照してください


⑥
②
①
③
DRB–0009
U2–type Bracket
(4ea)
DRB–0011
U4–type Bracket
(3ea)
DRB–0010
U3–type Bracket
(4ea)
DRB–0008
U1–type Bracket
(4ea)
DRB–0003
Foot (2ea)
④
⑤
HerkuleX
DRS-0101
Dongbu Robot

Humanoid DRC機能の使い方

コントローラーのオン

電源表示	残量が20%未満の時、点滅
バッテリー残量の表示	電源を入れ（左）ボタンを押して、バッテリー残量を確認できる→左側のLED3個で表示 最も低い時に1つ、中間の時に2個、最も高い時に3つ
Task進入	Modeボタンを押して、基本Taskを始める Navi key → Okボタンで希望のモードを選択

ロボットの動作

中間組立検査モード	Mode →（下）→ OK : 中間組立検査モードの組み立ての合間ごとに正しく組み立てているかどうかを確認 右腕、左腕、右足、左足、センサ
モーターの確認モード	Mode →（左）→ OK : モーターの確認モード – 実行すると、モーターの力がオフになる。モーターの順番に一つずつ選択される。選択されたモーターはLEDが点滅。 – （上）モーターID1ずつ増加、（下）モーターIDに1ずつ減少 – 該当するIDのモーターが存在しない場合はビープ音を鳴らす。
自律歩行モード	Mode →（上）→ OK : 自律歩行モード：ロボットが動きながら自ら動作 – 拍手をすると、拍手音の方向に回数だけ移動（動作中は反応しない） – 5秒間一定の反応がない時は基本動作を開始 – 基本動作：座った後立ち上がる、前後に歩く、方向を変更しながら歩く – 障害物の回避（PSDセンサが必要） – 倒れた時、自ら立ち上がる（ジャイロセンサが必要）
リモコン調整モード	Mode →（右）→ OK : リモコン調整モード リモコンの番号（0～9）と方向キー（上、下、左、右、止め）で定義されて動作する

Humanoid DRC機能の使い方

プログラムのダウンロード

	LED 動作	LED 表記
HerkuleX接続	HerkuleX Managerの実行時に点滅	Servo
DR-SIM/ Visual Logic 接続	DR-SIM/Visual Logicを使用して、編集する中に点滅する。 データやファームウェアをダウンロードする時にオン	Program
Taskをロボットで実行	Modeボタンを押して、Taskモードに切換後Taskの実行中にオン	EXEC
データ送信	外部へのデータ送信時点滅、taskの実行中はUser Spare空間	TX
データ受信	外部からのデータ受信時に点滅、taskの実行中はUser Spare空間	RX
ユーザーの割り当て		Spare
エラー発生	実行時にエラーが発生した場合、すべてのLEDが点滅して通知	全LED点滅

Humanoid DRC機能の使い方

ロボットの動作

検査モード	Mode →（左）→ OKを押して検査モードに入ります。 緑色のRX LEDがオンになった場合、（左）または（右）ボタンを押して、どのような点検をするかを選択します。 （左）: モーターの確認モード - モーターを接続するかどうかを一つずつ選択して組立状態を確認するモードです。 - モーターの確認モードに入る時、赤色のTX LEDがオンになります。 - 緑色のLEDがオンになると、選択したIDのモーターが中央の位置（512）に移動します。 その他のモーターはLEDがオフになり、トルクが解除されます。 -（上）（下）ボタンでIDを0～15まで変更し、各モーターの接続有無、組立の位置を確認してください。 最初に選択されたIDは0番（右肩）です。 -（上）ボタンを押すと、IDが1増加し、（下）ボタンを押すと、1減少します。 - 該当するIDのモーターが存在しない場合は警告ブザー音が鳴ります。 （右）: 中間組立検査モード - 腕、足などが確実に組み立てれているかをモジュールごとにテストすることができるモードです。 - 中間組立検査モードに入る時、青色のSpare LEDがオンになります。 - 選択されたモジュールに対応するモーターはまっすぐに伸ばした姿勢と曲げた姿勢を繰り返します。 -（上）（下）ボタンで、どの腕と足を移動するかを選択することができます。 - 左腕、右腕、左脚、右脚の順に選択されます。最初に選択されたモジュールは左腕です。 - 選択したモジュールに該当するモーターが接続されていない場合、接続されていないIDの数と同じ回数の ブザー音を鳴らして知らせます。
自律行動モード	Mode →（上）→ OKで自律行動モードにに入ります。 - ロボットが自らの行動で動作するモードです。 - 普段の状態の場合、ロボットは直進、前回り、左折、右折のいずれかをランダムに選択して動作します。 - 直進は10/20/30の歩行の中からランダムに選択され、左/右に曲がるときも12/24/36の歩行の中からランダムに 選択されます。 - 選択した動作を完了すると、しばらく止まってから次の動作を選択して動作します。 - PSDセンサがADCのポート1番に装備されている場合は障害物を認識して回避することができます。 - 動作中に障害物を認識すると、ロボットは後退して左折、左折、後回り後の左折のいずれかをランダム に選択して動作します。 - 後回り後、左折は直進を10歩以上している途中に障害物に出会った場合にのみ可能です。 - 左折を数回繰り返しても、障害物を避けることができなければ、再び後退し、左折します。 - 加速度センサがコントローラーに装着されている場合は転倒を検知して立ち上がることができます。 - 動作中に転倒した場合、自動的に方向を認識し、既存のモーションを停止して立ち上がるモーション を実行します。

Humanoid DRC機能の使い方

ロボットの動作

リモコン操縦モード	Mode ➔(右)➔ OKでリモコン操縦モードに入ります。 – リモコンでロボットの行動を操縦するモードです。 リモコンレシーバモジュールを接続する必要があります。 – 上:前進 – ハ:後退 – 左:左折 – 右:右折 – OK:停止 – 1 : 前回り – 2 : 後回り – 3 : 腕立て – 4:ボクシング – 5:前に立ち上がる(加速度センサが装備されていて、実際に後ろに倒れた状態のときに可能) – 6:後ろに立ち上がる(加速度センサが装備されていて、実際に前に倒れた状態のときに可能)
サウンドデモモード	Mode (下) OKでサウンドデモモードに入ります。 – 音の回数と方向の認識によってロボットが他の反応を見せるモードです。 – コントローラーのTX、RX、Spare LEDがオンになっている状態が音を認識している状態です。 – 音を認識している状態で、 – 1回音を入力した時:前回り、後回り、腕立て、ボクシングのいずれかのモーションをランダムに実行します。 – 2回音を入力した時:音の方向の腕を持ち上げて、ウェーブをします。例えば、左から一回、右から一回入力された場合、左腕、右腕の順でウェーブをします。 – 3回音を入力した時:音の方向に回転して10歩直進します。 例えば、左から音が出た場合、左折した後、直進します。 ※音の方向認識は周辺環境の騒音や壁による音の反射などに影響を受けやすく、正確に動作しない場合があります。また、拍手の音のように短く、大きな音を認識しやすいです。

インストール

DR-SIMの紹介

DR-SIMはロボットのモーションを簡単に編集するためのソフトウェアツールです。 Motion EditorといわれるDR-SIMは初心者も簡単に使用できます。 DR-SIMは強力なシミュレーション機能をサポートします。 ユーザーはこの機能を使用して、実際のロボットに適用されたモーションを事前にシミュレーション可能です。 これは既存のMotion Editorと異なる最大の点です。 また、DR-SIMの既存Motion Editorのように、 実際のロボットを操作して、モーションを作成する機能もサポートしています。 DR-SIMはタイムラインを提供し、ビデオエディタのような直感的なUIを提供しています。ユーザーは時間に応じてロボットのモーションを作る事が可能です。頭側のLEDの編集やサウンドも、モーションのようにタイムラインに挿入して編集することができ、マルチメディア的な編集が可能です。最後にDR-SIMはマルチメディアの編集をサポートしています。ロボット頭部のLEDとタイムライン上のサウンド機能を利用して、ロボットのモーションのマルチメディア効果を追加してください。

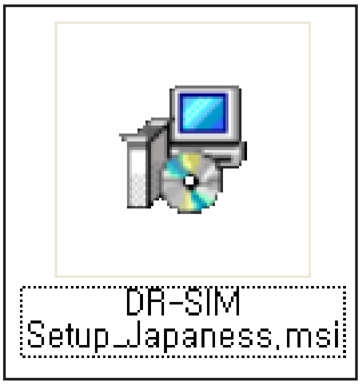
- システム要件
- 最低インテルペンティアム800Mhz以上
- Windows XP、Windows Vista、Windows7
- 最低256 MB RAM
- ハードディスクの設置スペース300 MBが必要
- USB Port
- Macintosh（対応予定）

PART
05

ソフトウェア：DR-SIM

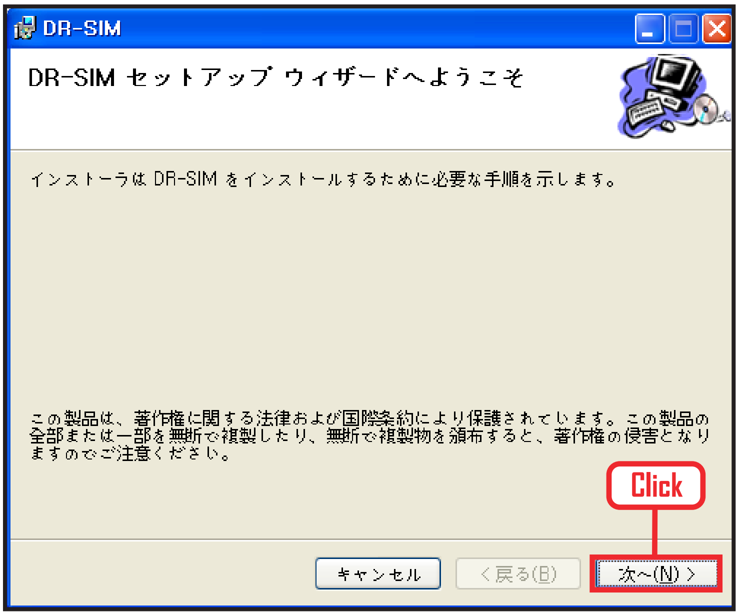
インストール

インストールから実行まで下の手順で実行してください


DR-SIM
Setup_Japaness.msi

01 インストールファイル

インストールファイルをクリックします。



DR-SIM

DR-SIM セットアップ ウィザードへようこそ

インストーラは DR-SIM をインストールするために必要な手順を示します。

この製品は、著作権に関する法律および国際条約により保護されています。この製品の全部または一部を無断で複製したり、無断で複製物を頒布すると、著作権の侵害となりますのでご注意ください。

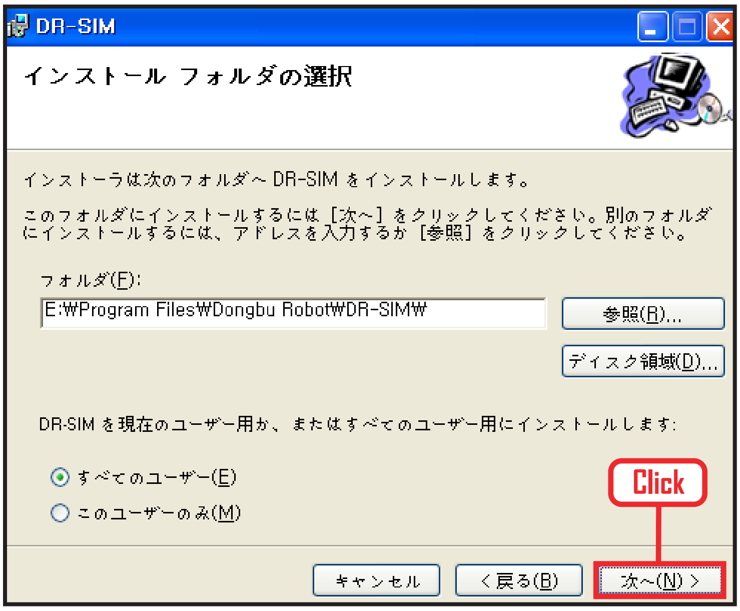
Click
キャンセル
< 戻る(B)
次へ(N) >

02 インストールウィザードの開始

“次へ”ボタンをクリックします。

03 インストールフォルダを選択

“次へ”ボタンをクリックします。



インストール フォルダの選択

インストーラは次のフォルダへ DR-SIM をインストールします。

このフォルダにインストールするには［次へ］をクリックしてください。別のフォルダにインストールするには、アドレスを入力するか［参照］をクリックしてください。

フォルダ(F):
E:₩Program Files₩Dongbu Robot₩DR-SIM₩

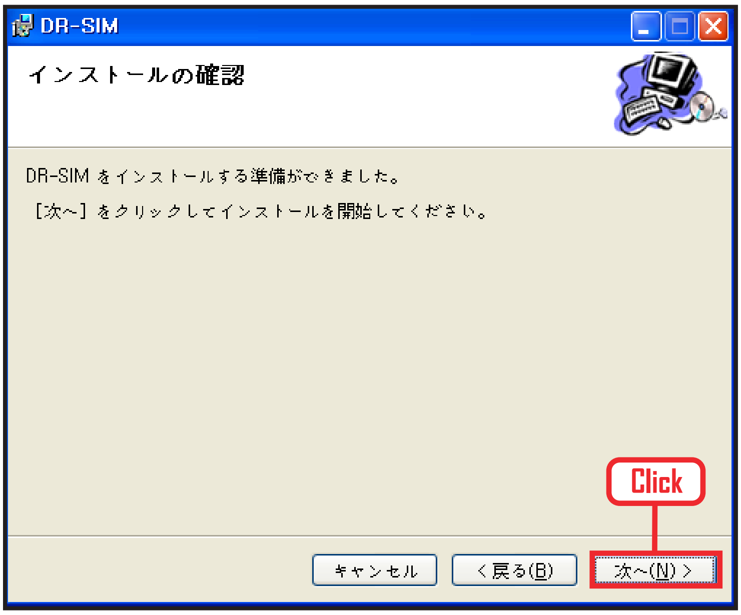
DR-SIM を現在のユーザー用か、またはすべてのユーザー用にインストールします:

● すべてのユーザー(E)
○ このユーザーのみ(M)

DR-SIM
参照(R)...
ディスク領域(D)...
Click
キャンセル
< 戻る(B)
次へ(N) >

04 インストール確認

“次へ”ボタンをクリックします。



インストールの確認

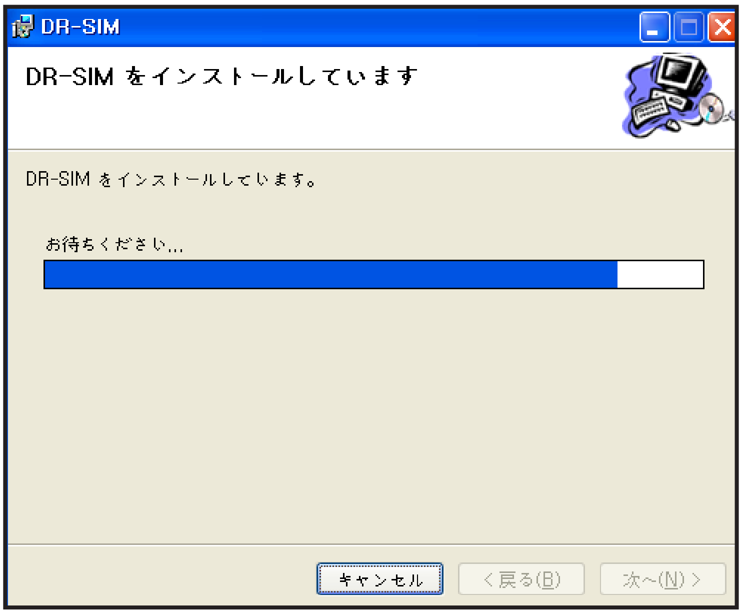
DR-SIM をインストールする準備ができました。

［次へ］をクリックしてインストールを開始してください。

DR-SIM
Click
キャンセル
< 戻る(B)
次へ(N) >

05 インストールの開始

インストールを開始します。プログレスバーが終わるまでお待ちください

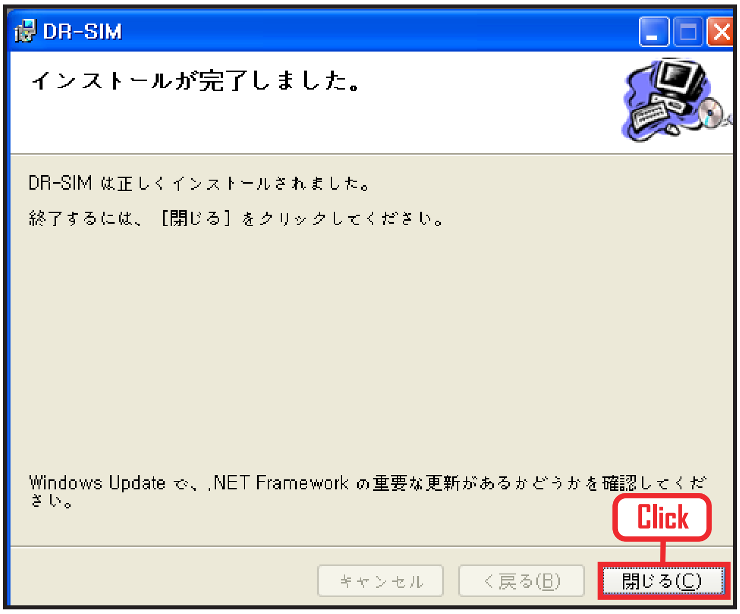


DR-SIM をインストールしています

DR-SIM をインストールしています。

お待ちください...

DR-SIM
キャンセル
< 戻る(B)
次へ(N) >

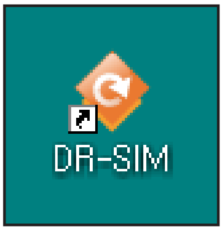


インストールが完了しました。

DR-SIM は正しくインストールされました。

終了するには、［閉じる］をクリックしてください。

Windows Update で、.NET Framework の重要な更新があるかどうかを確認してください。



06 インストール完了の選択

“閉じる”をクリックしてください
ソフトウェアのインストールが完了しました。

07 実行ファイルの確認

デスクトップのショートカットアイコンとWindowsのスタート > すべてのプログラム > Dongbu Robot> DR-SIMにある実行ファイルを確認してください。
実行ファイルをクリックすると、プログラムが実行されます。

インストールができなかった場合、Microsoft、Net Frame work3.5をインストールし、もう一度試してください。

PART
05

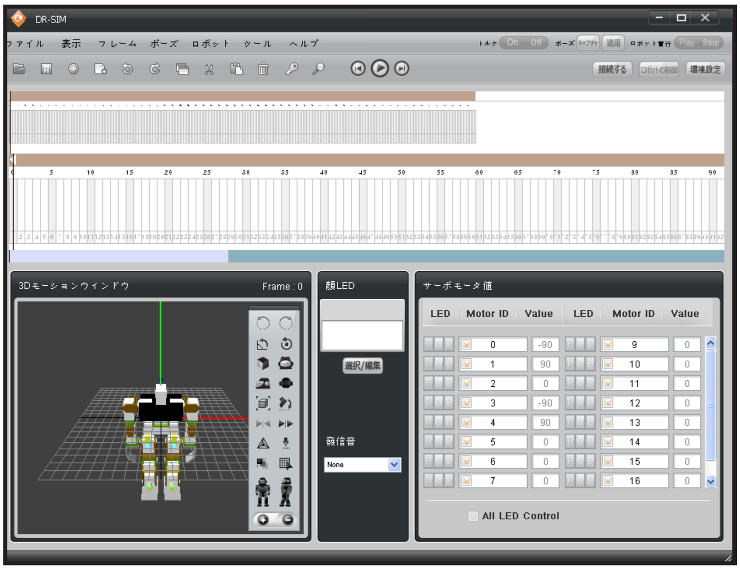
ソフトウェア：DR-SIM

Hello DR-SIM

ロボットのモーションを作成する一番目の例です。DR-SIMを利用して簡単なモーションを編集した後、これをシミュレーションします。ロボットに接続して編集されたモーションファイルをダウンロードします。そして、ロボットで実行されるモーションを確認してください。

00 実行

DR-SIMの実行アイコンをクリックしてプログラムを実行します。

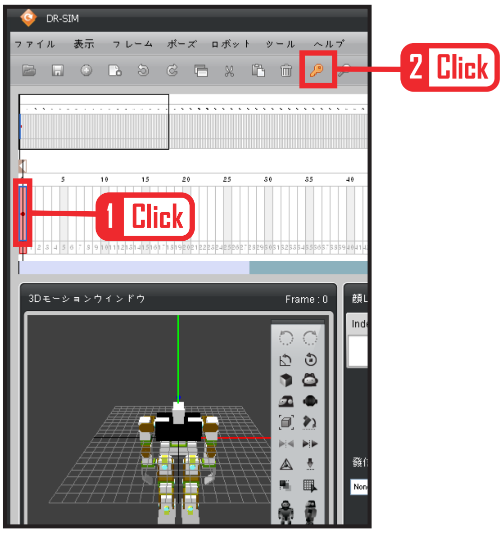

DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
接続する
環境設定
Play
Stop
3Dモーションウィンドウ
Frame : 0
顔LED
選択/編集
発信音
None
サーボモータ値

LED	Motor ID	Value	LED	Motor ID	Value
R G B	0	-90	R G B	9	0
R G B	1	90	R G B	10	0
R G B	2	0	R G B	11	0
R G B	3	-90	R G B	12	0
R G B	4	90	R G B	13	0
R G B	5	0	R G B	14	0
R G B	6	0	R G B	15	0
R G B	7	0	R G B	16	0

☐ All LED Control

01 全画面表示

DR-SIMの全画面表示です。
中央にタイムラインがあり下端にモーション編集があります。主に画面中央のタイムラインと下端の3Dモーションウィンドウでモーションを編集します。

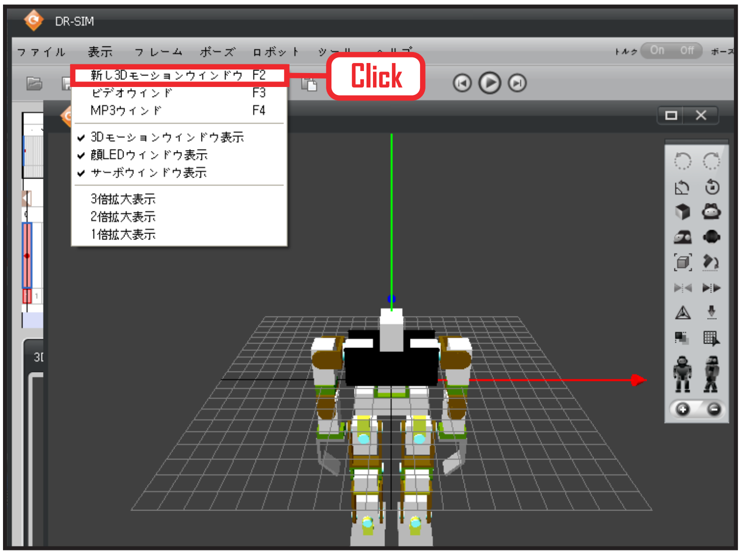

DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
2 Click
1 Click
3Dモーションウィンドウ
Frame : 0

02 基本姿勢

ロボットの動作の開始点や最初のフレームにロボットの基本動作を挿入します。

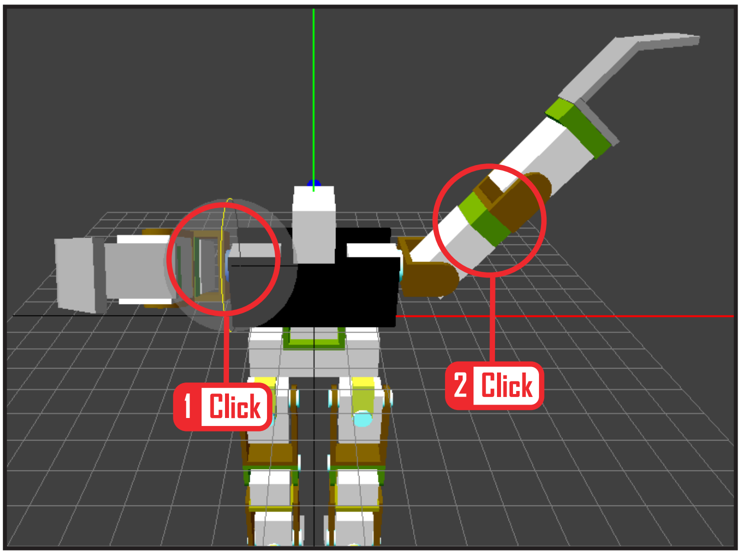
3Dモーションウィンドウの動作を基本姿勢に合わせる ➔ 最初のフレームをクリック ➔ 上のキーフレームを挿入する（キーアイコン）をクリック
基本姿勢が最初のキーフレームに挿入されます。

3Dモーションウィンドウの姿勢が基本姿勢でなければ右のツールバーで基本姿勢にしてから実行します（ショートカットキーAlt+ I）


DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
新しい3Dモーションウィンドウ F2
ビデオウィンドウ F3
MP3ウィンドウ F4
✓ 3Dモーションウィンドウ表示
✓ 顔LEDウィンドウ表示
✓ サーボウィンドウ表示
3倍拡大表示
2倍拡大表示
1倍拡大表示
Click

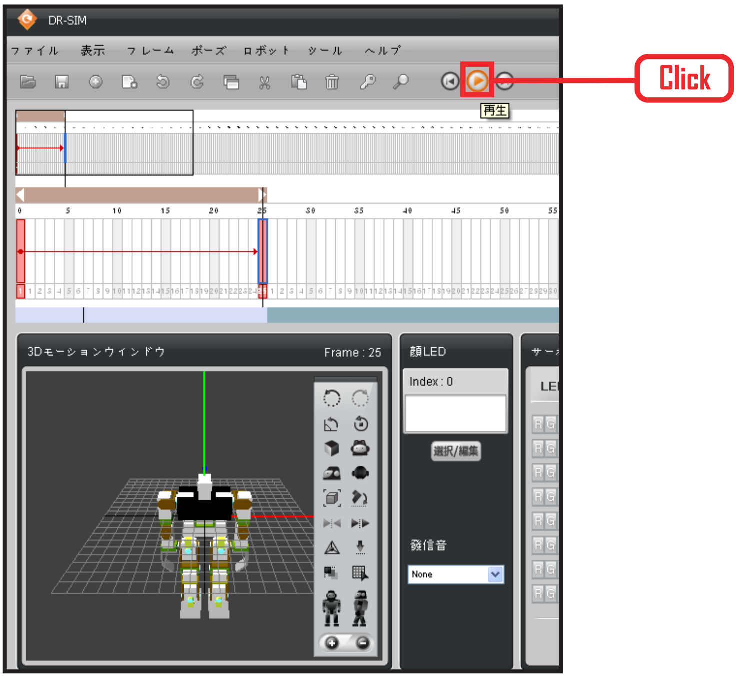
03 3Dウィンドウ

3Dウィンドウを拡大表示するためにはメニュー>表示>新しい3Dウィンドウの表示をクリックします。
新しい3Dのウィンドウが表示されたら、画面をドラッグして拡大します。


1 Click
2 Click

04 モーションの編集

ロボットの関節をクリックすると、関節の移動線が黄色の実線で表示されます。
その線に沿ってマウスの左を押しながらドラッグします。
ロボットの左手は左上側に上げて、ロボットの右手には前上側に上げます。

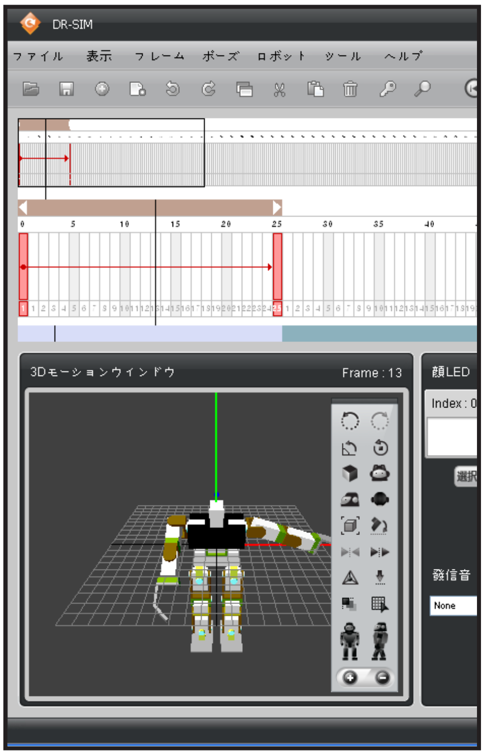

DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
再生
Click
3Dモーションウィンドウ
Frame : 25
顔LED
Index : 0
選択/編集
発信音
None

05 編集したモーションを挿入する

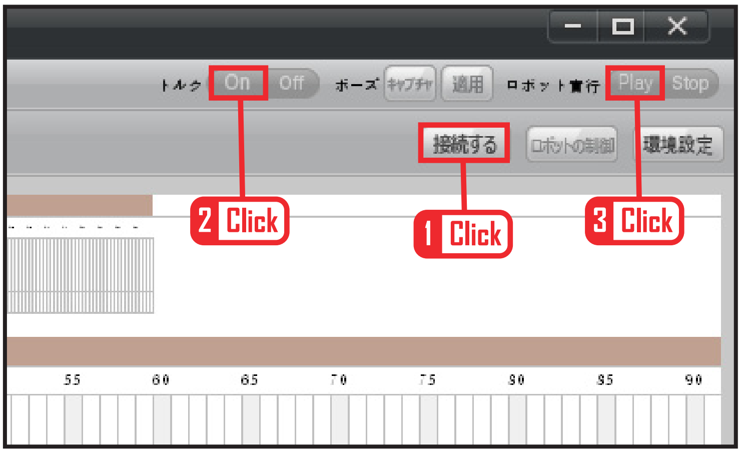
3Dウィンドウで編集されたロボットのモーションをタイムフレームに挿入します。
挿入後、上部の“▶”ボタンをクリックします。
ロボットの動作シミュレーションを3Dウィンドウで確認できます。

06 挿入フレームの中間確認

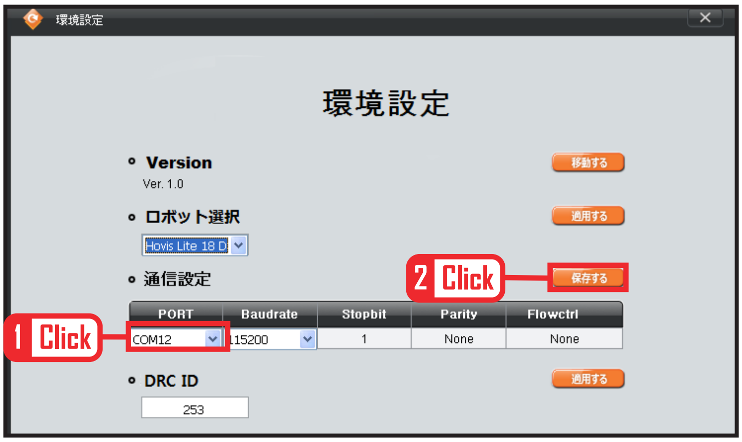
基本モーションと編集されたモーションの中間タイムライン]をクリックします。
二つのフレーム間の中間動作を確認できます。


DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
3Dモーションウィンドウ
Frame : 13
顔LED
Index : 0
選択
發信音
None

07 ロボットとの接続


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット實行
Play
Stop
接続する
ロボットの制御
環境設定
2 Click
1 Click
3 Click
55
60
65
70
75
80
85
90

PCやノートパソコンのUSB端子にUSB to Serialコンバータを用いてロボットに接続します。
“接続”アイコンをクリックすると、ロボットと接続されます。
接続できない場合はCOM portを確認してください。
“トルクON”ボタンをクリックして、ロボットのトルクの状態を確認して動かないとトルクオンの状態です。
“ロボット実行Play”ボタンをクリックすると、ロボットが編集モーションのように動きます。
これで、最初のモーション編集とロボットでの実行が完了しました。


環境設定
移動する
適用する
保存する
適用する
1 Click
2 Click

環境設定

- **Version**
  Ver. 1.0
- **ロボット選択**
  Hovis Lite 18 D
- **通信設定**

PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

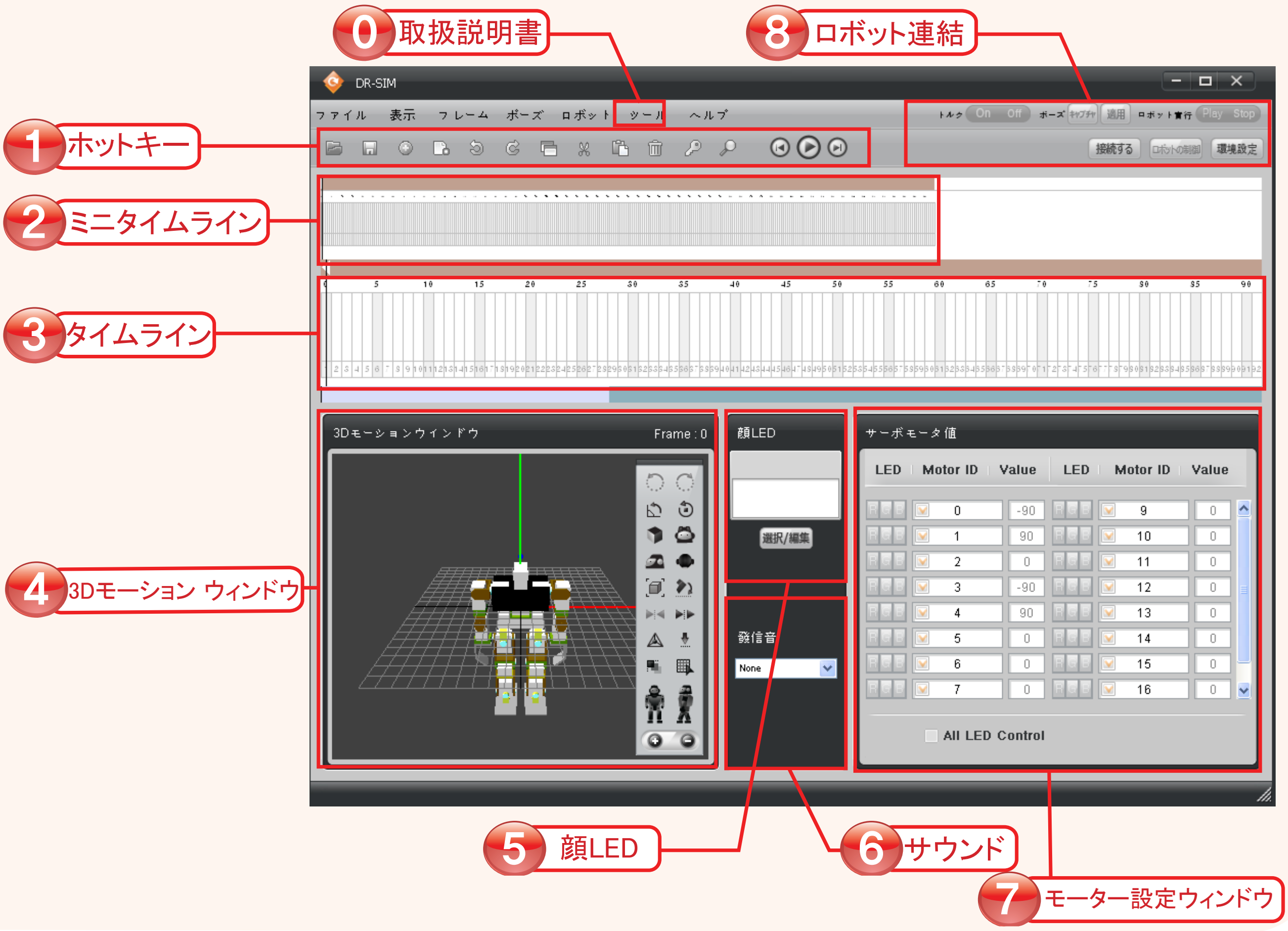
- **DRC ID**
  253

08 参照：設定COMポートの設定

通信設定でCOM2を選択して保存します。
ロボットとPCのソフトウェアが接続されます。

ソフトウェア：DR-SIM

ユーザーインターフェイス


0 取扱説明書
8 ロボット連結
1 ホットキー
2 ミニタイムライン
3 タイムライン
4 3Dモーション ウィンドウ
5 顔LED
6 サウンド
7 モーター設定ウィンドウ
DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
接続する
ロボットの制御
環境設定
3Dモーションウィンドウ
Frame : 0
顔LED
選択/編集
發信音
None
サーボモータ値
All LED Control

LED	Motor ID	Value	LED	Motor ID	Value
	0	-90		9	0
	1	90		10	0
	2	0		11	0
	3	-90		12	0
	4	90		13	0
	5	0		14	0
	6	0		15	0
	7	0		16	0

☐ All LED Control

0 使用説明書：DR-SIMの詳細な使い方が示されています。F1を押すと、ヘルプのショートカットが実行されます。

1 ショートカットキー：使用頻度の高いメニューを集めました。シミュレーションの再生、キーフレームの挿入などがあります。

2 ミニタイムライン：タイムフレームの全体のアウトラインを見ることができます。

3 タイムライン：ロボットのモーションを編集した後、時間ごとにモーションを配置できます。

4 3Dモーションウィンドウ：シミュレーションウィンドウで編集したロボットのモーションを仮想的に動かすくことができます。

5 顔LED：ロボットの顔のLEDを編集できるウィンドウです。編集後、タイムフレームに挿入します。

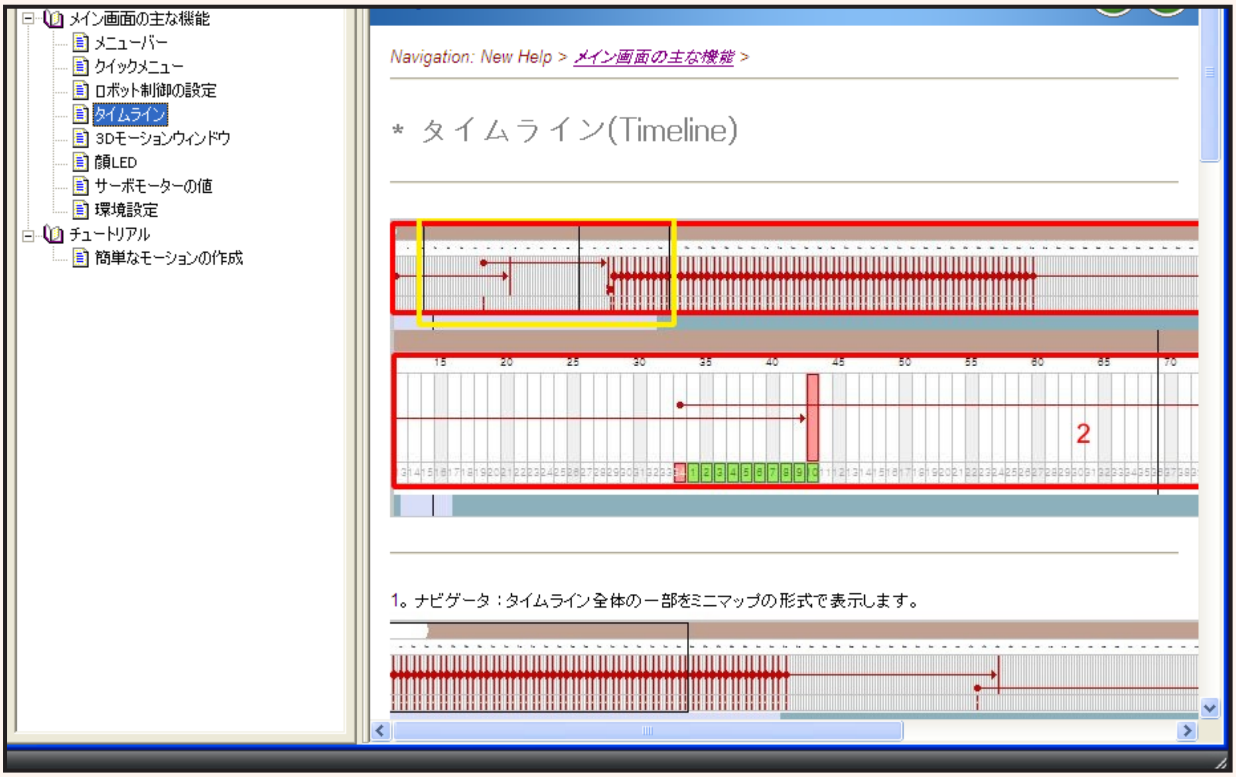
6 サウンド：保存されたサウンドを指定できます。サウンドを選択し、タイムフレームに挿入します。

7 モーターの設定画面：ロボットに装着されたすべてのモーターIDの値とLEDを設定できます。

8 ロボットの接続：ロボットと接続するためのショートカットです。編集したモーションファイルをロボットにダウンロードしたり、実際のロボットのポーズをキャプチャする用途で使われます。

ヘルプの使い方

メニューから[ツール] > [ヘルプ]をクリックすると、以下のようなヘルプのポップアップウィンドウが表示されます。DR-SIMを使用するためには先ずヘルプをお読みいただいた上で使用することをお勧めします。（上部[ヘルプ]をクリック > [目次]をクリック > 左側のメニューの[タイムライン]をクリック ➜ 下の図が表示されます）


メイン画面の主な機能
メニューバー
クイックメニュー
ロボット制御の設定
タイムライン
3Dモーションウィンドウ
顔LED
サーボモーターの値
環境設定
チュートリアル
簡単なモーションの作成
2

Navigation: New Help > メイン画面の主な機能 >

＊ タイムライン(Timeline)

1。ナビゲータ：タイムライン全体の一部をミニマップの形式で表示します。

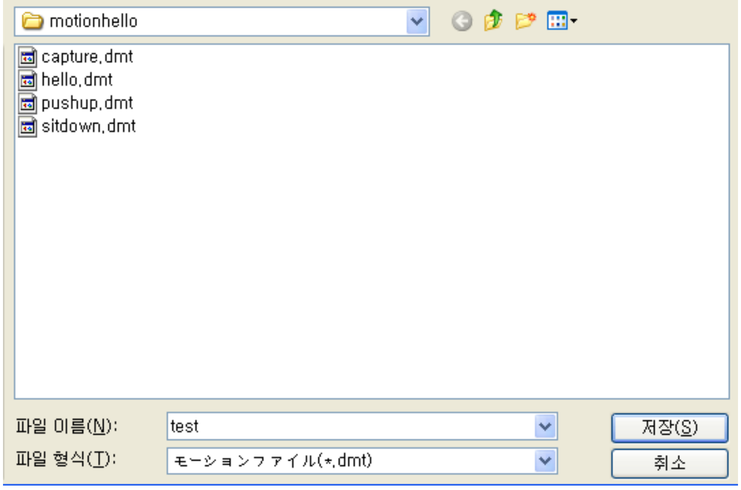
- 接続を設定する：COMポートを接続する方法について説明します。
- メイン画面の主な機能：プログラムの機能の使い方を説明します。
  - メニューバー
  - クイックメニュー
  - ロボット制御の設定
  - タイムライン
  - 3Dモーションウィンドウ
  - 顔LED
  - サーボモーター値
  - 環境設定
- チュートリアル
  - 簡単なモーションの作成：基本モーションの作成例を説明します。
  - 様々なモーションサンプルのソースはWebサイトで提供しています。

ダウンロードする

編集したロボットモーションはファイルに保存されます。保存されたロボットのモーションファイルを一度にDRCコントローラーにダウンロードすることができます。（この時、DRCコントローラーにあった以前のロボットのモーションは削除されます。）ダウンロードされたロボットのモーションファイルはDRCコントローラーにダウンロードされた順に番号が付与されます。設定された番号はDR-Visual Logic（Task Editor）のプログラムで読み込んで使用します。

ロボットのモーションを保存

編集されたロボットのモーションは［ファイル］>［名前を付けて保存］メニューをクリックしてタイトルを付けた後、保存フォルダに保存します。



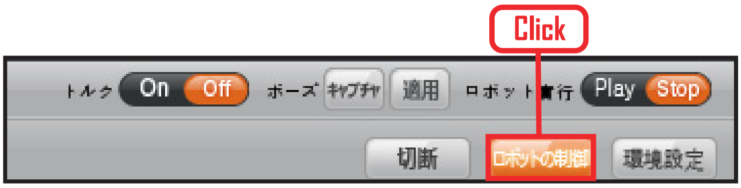
motionhello

capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt

파일 이름(N): test
파일 형식(T): モーションファイル(*.dmt)

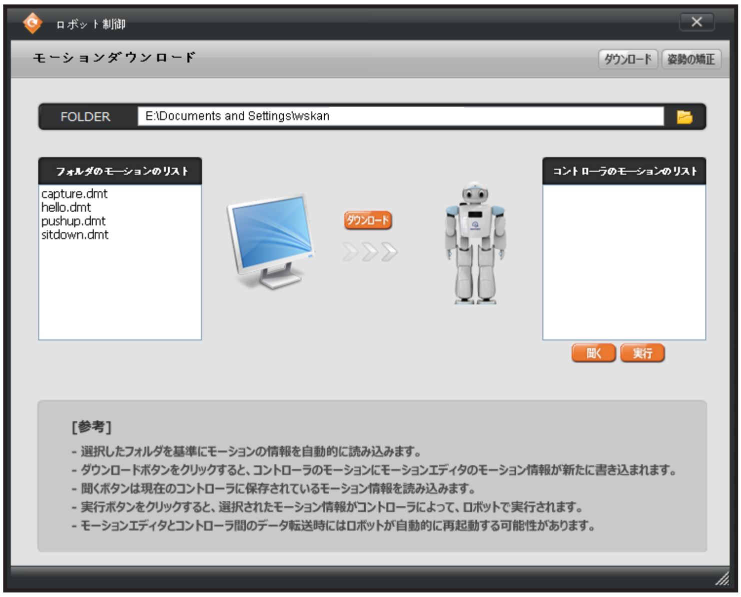
01 ロボットの制御

ロボットとDR-SIMを接続した状態で"ロボット制御"アイコンをクリックします。


Click
トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの制御
環境設定

02 ロボットの制御ウィンドウ

モーション情報のダウンロードのポップアップが表示されます。モーションが保存されたフォルダへのパスがあり、左側にはPCに保存されたモーションのリストがあります。コントローラーにダウンロードすると、ロボットのモーションが、右側に番号順にリストアップされます。


ロボット制御
モーションダウンロード
ダウンロード
姿勢の矯正
フォルダのモーションのリスト
コントローラのモーションのリスト
ダウンロード
開く
実行

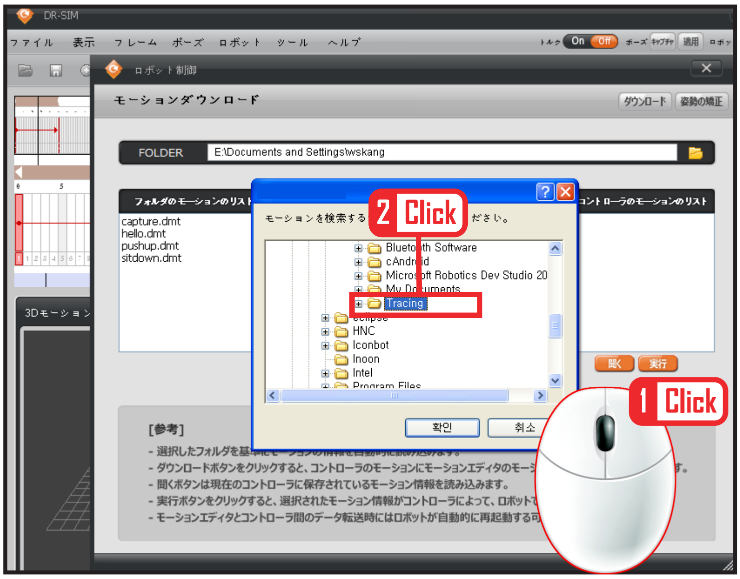
FOLDER E:\Documents and Settings\wskan

capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt

[参考]
- 選択したフォルダを基準にモーションの情報を自動的に読み込みます。
- ダウンロードボタンをクリックすると、コントローラのモーションにモーションエディタのモーション情報が新たに書き込まれます。
- 開くボタンは現在のコントローラに保存されているモーション情報を読み込みます。
- 実行ボタンをクリックすると、選択されたモーション情報がコントローラによって、ロボットで実行されます。
- モーションエディタとコントローラ間のデータ転送時にはロボットが自動的に再起動する可能性があります。

03 保存フォルダを開く

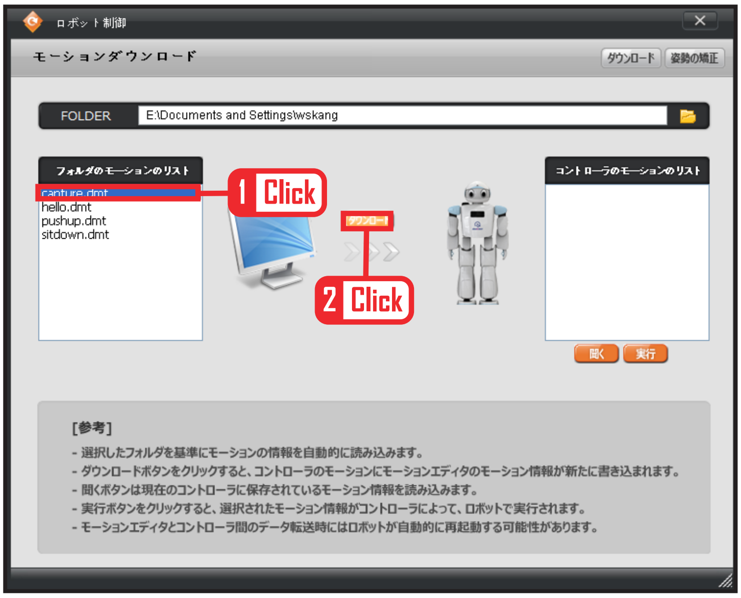
フォルダパスの右側をクリックすると、フォルダの[参照]ポップアップウィンドウが表示されます。保存したフォルダを選択します。


DR-SIM
ファイル 表示 フレーム ポーズ ロボット ツール ヘルプ
トルク On Off ポーズ キャプチャ 適用
ロボット制御
モーションダウンロード
ダウンロード
姿勢の矯正
FOLDER
E:\Documents and Settings\wskang
フォルダのモーションのリスト
capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt
コントローラのモーションのリスト
モーションを検索する ださい。
Bluetooth Software
cAndroid
Microsoft Robotics Dev Studio 20
My Documents
Tracing
eclipse
HNC
Iconbot
Inoon
Intel
Program Files
확인
취소
開く
実行
3Dモーション
2 Click
1 Click

[参考]
- 選択したフォルダを基
- ダウンロードボタンをクリックすると、コントローラのモーションにモーションエディタのモーシ す。
- 開くボタンは現在のコントローラに保存されているモーション情報を読み込みます。
- 実行ボタンをクリックすると、選択されたモーション情報がコントローラによって、ロボットで
- モーションエディタとコントローラ間のデータ転送時にはロボットが自動的に再起動する可

04 モーションリスト

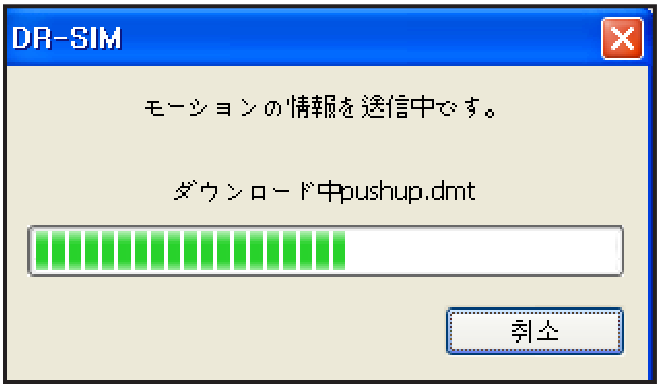
選択したフォルダにあるモーションのリストが表示されます。リストにカーソルを置き、ダウンロードアイコンをクリックします。


ロボット制御
モーションダウンロード
ダウンロード
姿勢の矯正
FOLDER
E:\Documents and Settings\wskang
フォルダのモーションのリスト
capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt
1 Click
ダウンロード
2 Click
コントローラのモーションのリスト
開く
実行

[参考]
- 選択したフォルダを基準にモーションの情報を自動的に読み込みます。
- ダウンロードボタンをクリックすると、コントローラのモーションにモーションエディタのモーション情報が新たに書き込まれます。
- 開くボタンは現在のコントローラに保存されているモーション情報を読み込みます。
- 実行ボタンをクリックすると、選択されたモーション情報がコントローラによって、ロボットで実行されます。
- モーションエディタとコントローラ間のデータ転送時にはロボットが自動的に再起動する可能性があります。

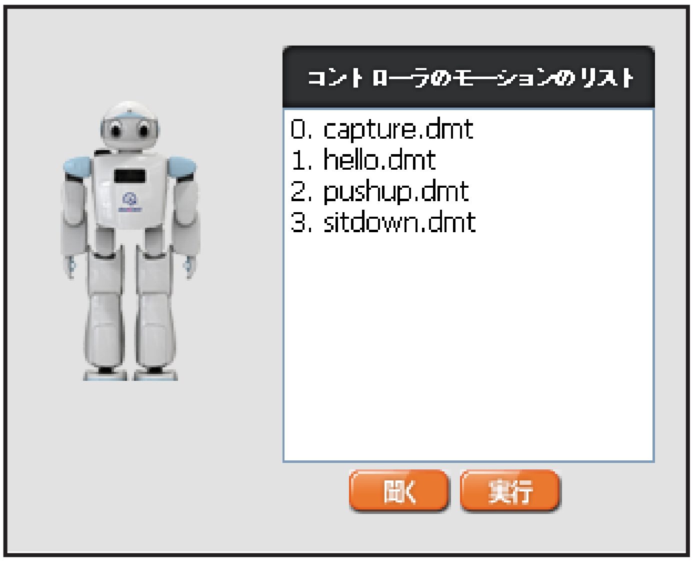
05 ダウンロード

ロボットのモーションが一つずつダウンロードされます。



モーションの情報を送信中です。

ダウンロード中pushup.dmt



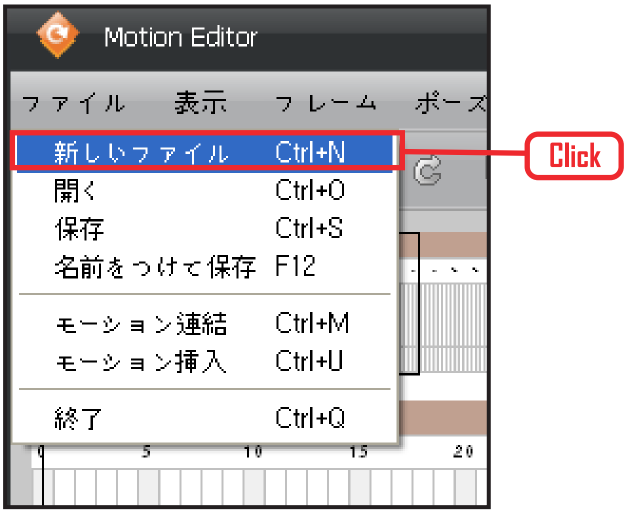
0. capture.dmt
1. hello.dmt
2. pushup.dmt
3. sitdown.dmt

06 コントローラーのモーションリスト

すべてのモーションが全部ダウンロードされると、右側のコントローラーのモーションリストに0番からモーションがリストアップされます。
番号はDR-Visual Logic（Task Editor）でのプログラミング時にインデックスで呼び出すことができます。
これにより、ロボットのモーションのダウンロードが完了しました。

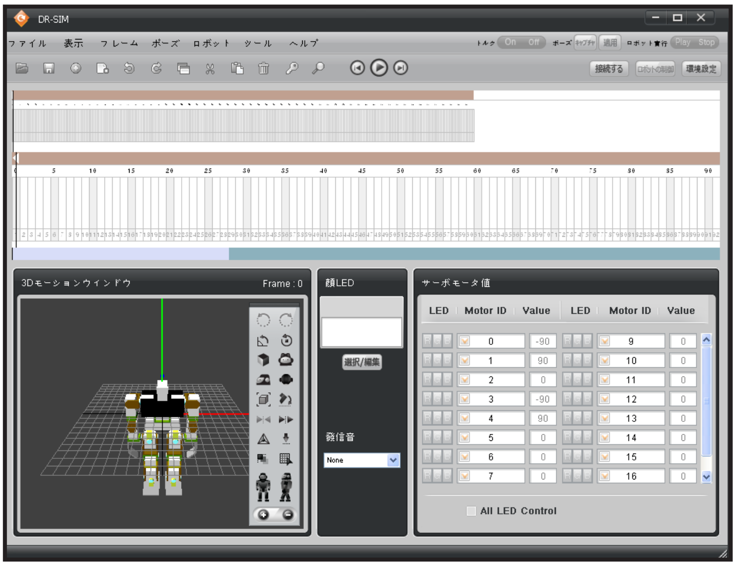
モーションの作成

モーションの作成は大きく二つの方法があります。 3Dモーションウィンドウで関節ごとにクリックして移動線にモーションを作成する方法と、ロボットのハードウェア自体に動作を作成してキャプチャーする方法があります。この二つの方法を活用し、簡単なモーションを作成しましょう。


Motion Editor
ファイル
表示
フレーム
ポーズ
新しいファイル Ctrl+N
開く Ctrl+O
保存 Ctrl+S
名前をつけて保存 F12
モーション連結 Ctrl+M
モーション挿入 Ctrl+U
終了 Ctrl+Q
Click
0
5
10
15
20

01 新規

ファイル > 新規 をクリックします。

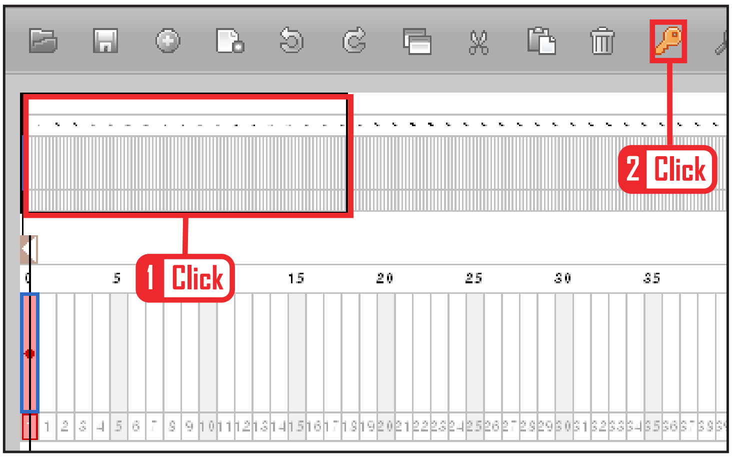

DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
3Dモーションウィンドウ
Frame : 0
顔LED
選択/編集
サーボモータ値
LED
Motor ID
Value
All LED Control

02 新しいモーション編集ウィンドウ

以前の編集ウィンドウが消えて、新しいロボットの3Dモーションウィンドウが表示されます。

03　最初のフレーム

ロボットの基本姿勢を最初のフレームに挿入します。
最初のフレームをクリックし、［キーフレームの挿入］をクリックします。

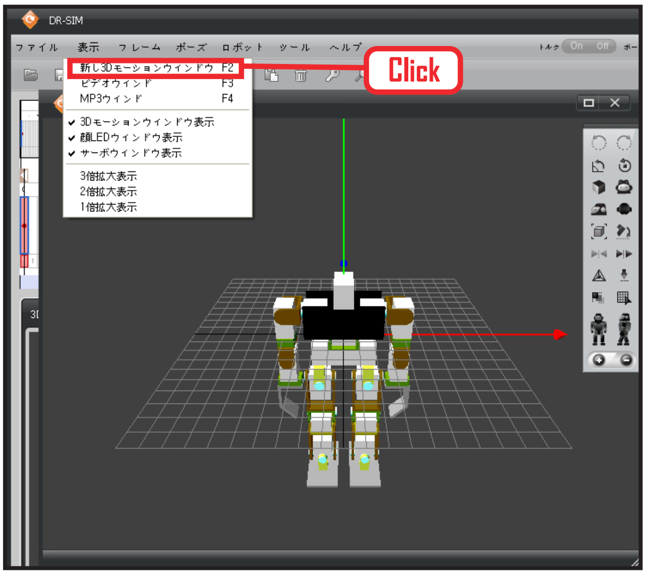

1 Click
2 Click
5
15
20
25
30
35

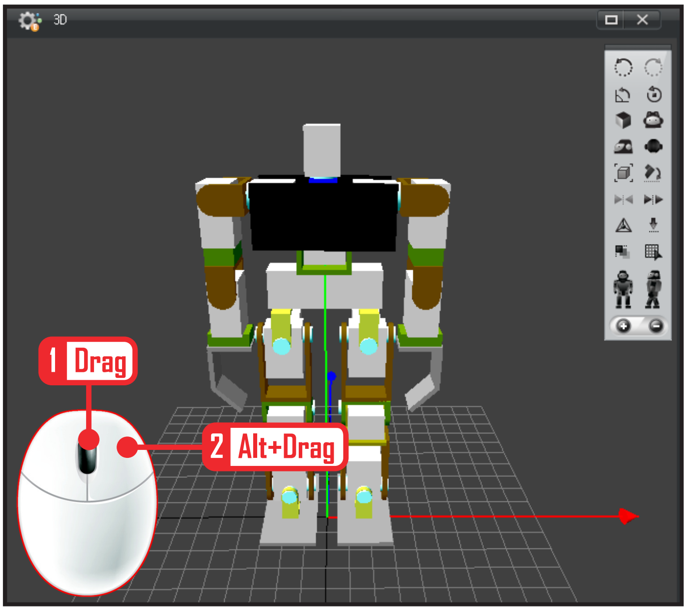
04　新しいモーションウィンドウ

画面でモーションを編集するためにはモーションウィンドウを拡大する必要があります。モーションウィンドウのみを別に開き拡大して編集します。 3Dモーションのウィンドウは最大3個まで同時に開くことができます。それぞれ上、下、側面の状態に設定し、編集することができます。

表示 ＞ 新しい3Dモーションウィンドウ をクリックします。

ポップアップモーションウィンドウが表示されたら、右下端の部分を右クリックしてつかんでドラッグして画面を拡大します。

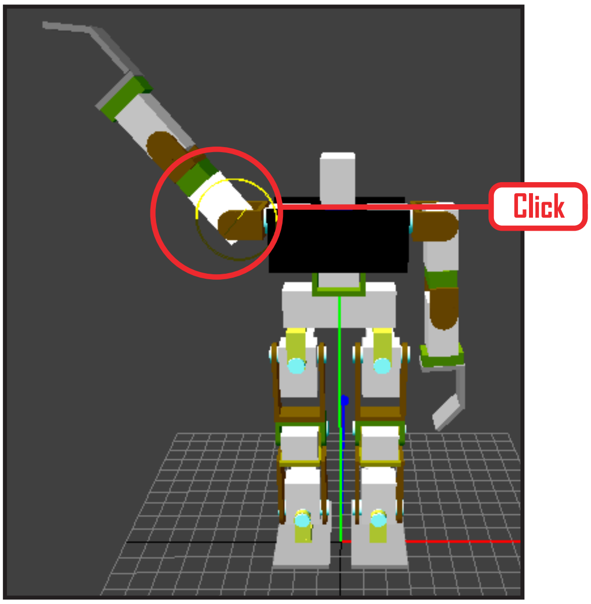

DR-SIM
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
新し3Dモーションウィンドウ　F2
ビデオウィンド　F3
MP3ウィンド　F4
✓ 3Dモーションウィンドウ表示
✓ 顔LEDウィンドウ表示
✓ サーボウィンドウ表示
3倍拡大表示
2倍拡大表示
1倍拡大表示
Click



05 ロボットの拡大

編集ウィンドウ中のロボットを拡大します。
空いている空間でマウスをクリックし、マウスホイールでロボットを拡大縮小することができます。編集しやすくロボットを拡大します。

角度を調節したい場合はマウスの右ボタンを押した状態でドラッグします。ロボットの位置はドラッグ(Shift+マウスの右ボタン)して調整します。

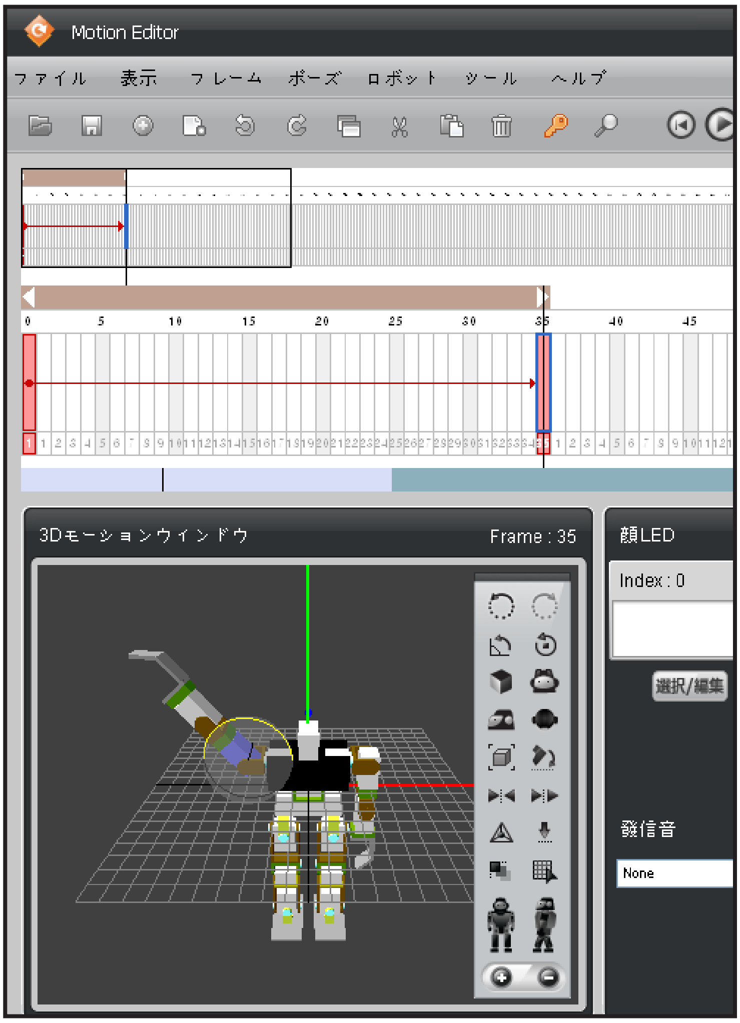


06 腕のモーション編集

腕をあげます。肩の部分をクリックします。クリックすると、黄色で移動線が表示されます。移動線に沿ってドラッグすると、腕が上がります。

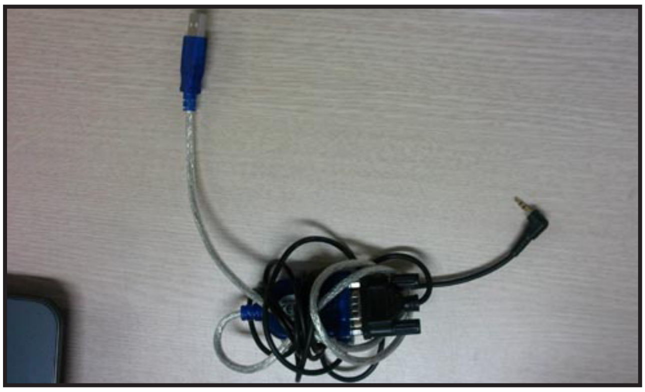
07 キーフレームの挿入

腕を上げたモーションをフレームに挿入します。挿入したい部分のフレームをクリックし、キーのアイコンをクリックして、モーションを挿入します。


Motion Editor
ファイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
0
5
10
15
20
25
30
35
40
45
3Dモーションウインドウ
Frame : 35
顔LED
Index : 0
選択/編集
發信音
None

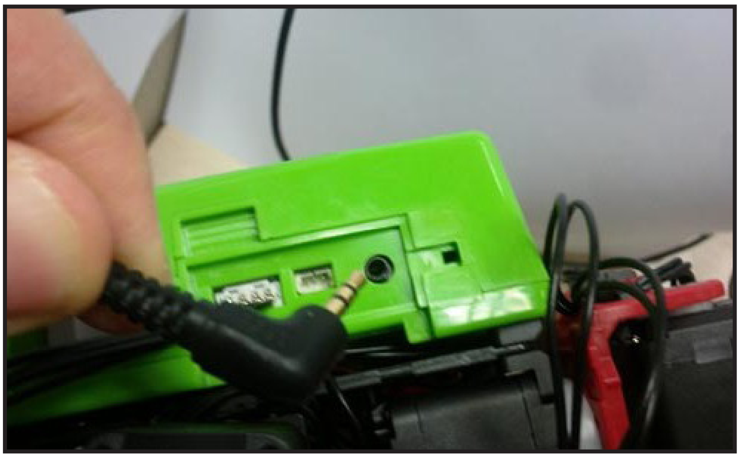
08−1 USB to Serial コンバータ

ロボットとの接続を開始します。
PCのUSB Portから、ロボットと接続できます。



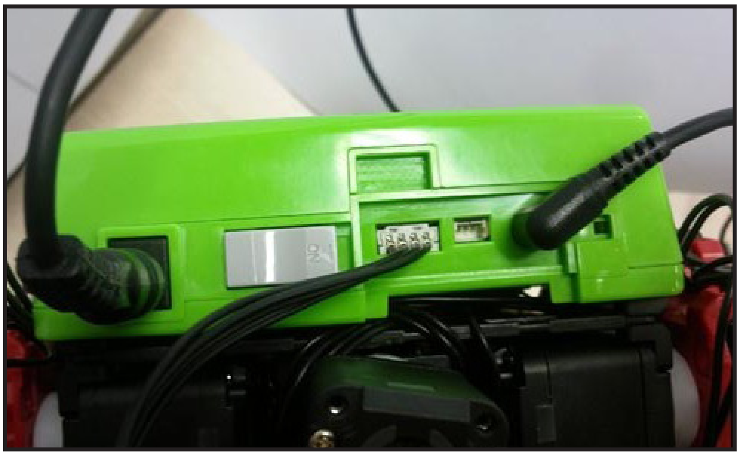
08−2 USB port

PCは裏側にSerial Portがあるので、すぐに接続できますが、ノートパソコンのようにSerial Portがない時はUSB to Serial コンバータを使用して接続します。



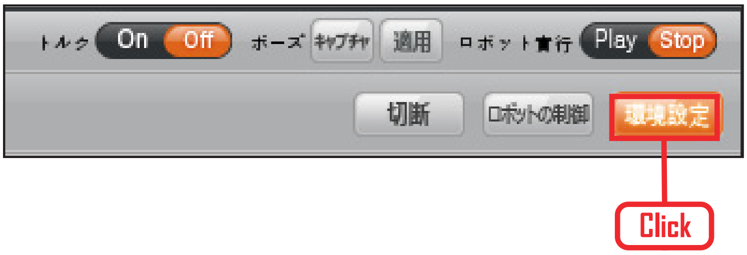
08−3 ロボットとの接続

RS232cのオーディオジャックをロボットに接続します。



08−4 ロボット Port

DRCコントローラーの上部を見ると、PCと接続するSerial Portと頭部のサーボモーターPort、電源のPortがあります。
すべてを接続した写真です。

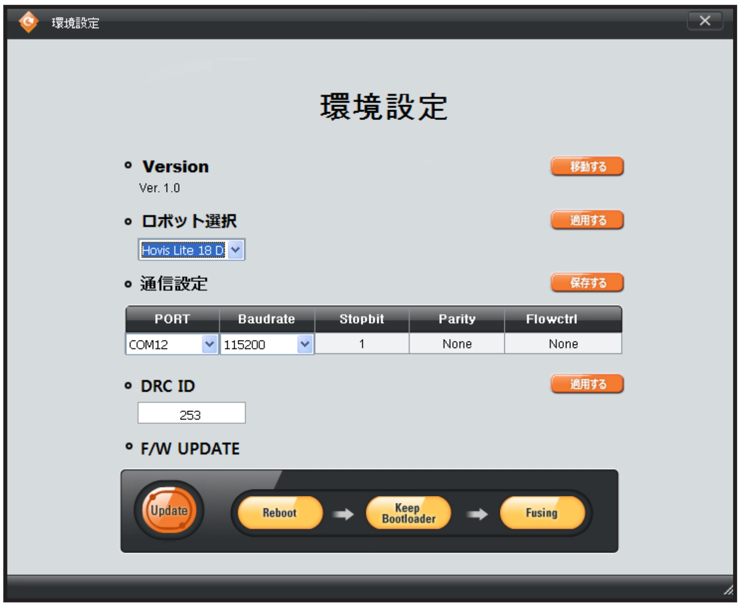

トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの制御
環境設定
Click

08−5 ロボットの接続ボタン

ロボットと接続するメニューはDR-SIMの右上にあります。
COM Portを設定するための環境設定をクリックします。

09 環境設定

DR-SIM version、ロボットの選択、通信設定などができる設定ウィンドウです。


環境設定
移動する
適用する
保存する
適用する
Update
Reboot
Keep Bootloader
Fusing

環境設定

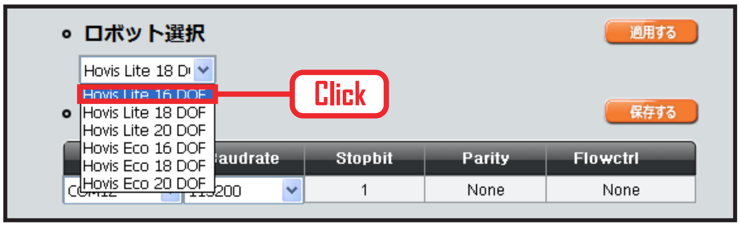
- Version
  Ver. 1.0
- ロボット選択
  Hovis Lite 18 D
- 通信設定

PORT	Baudrate	Stopbit	Parity	Flowctrl
COM12	115200	1	None	None

- DRC ID
  253
- F/W UPDATE

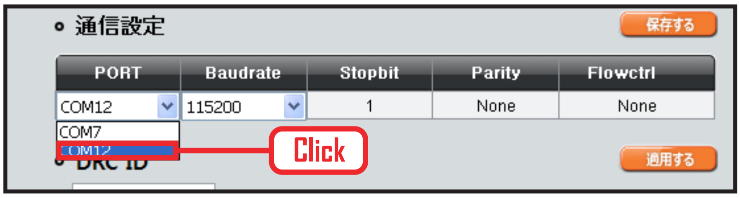
10 ロボットの選択

DR-SIMは合計6つのタイプのヒューマノイドを提供しています。その中で最も基本的なものは16DOFヒューマノイドです。ユーザーが組み立てたロボットの形状を選択します。
Hovis Lite16 DOFを選択します。


ロボット選択
適用する
保存する
Click
Hovis Lite 16 DOF
Hovis Lite 18 DOF
Hovis Lite 20 DOF
Hovis Eco 16 DOF
Hovis Eco 18 DOF
Hovis Eco 20 DOF

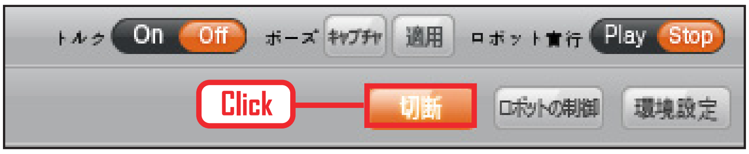
11 COM Port 選択

“PORT”に接続できるCOM Port番号が表示されます。その中の一つを選択してください。接続できない場合はハードウェアのプロパティで利用できるCOM Port番号を確認します。


通信設定
保存する
PORT
Baudrate
Stopbit
Parity
Flowctrl
COM12
115200
1
None
None
COM7
COM12
Click
DRC ID
適用する

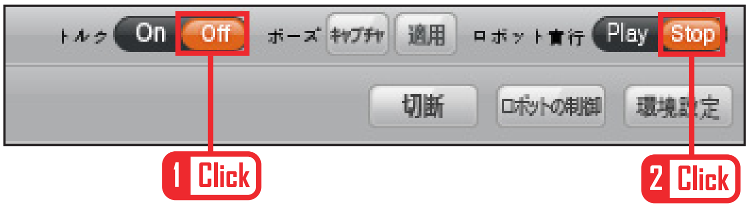
12 接続する

接続するアイコンをクリックします。

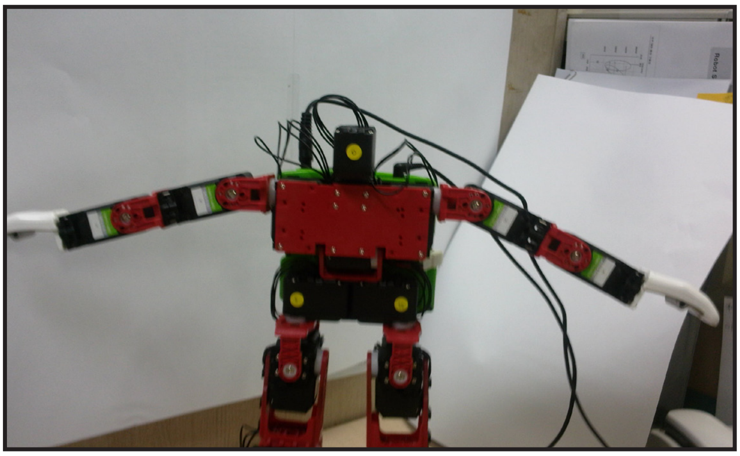

トルク On Off
ポーズ キャプチャ 適用
ロボット実行 Play Stop
Click
切断
ロボットの制御
環境設定

13 接続

接続が完了すると、左図のようにトルクボタンが有効になります。
ロボットを動作させるためには”トルクOn”ボタンをクリックします。現在のモーションを再生するためにはplayボタンをクリックします。

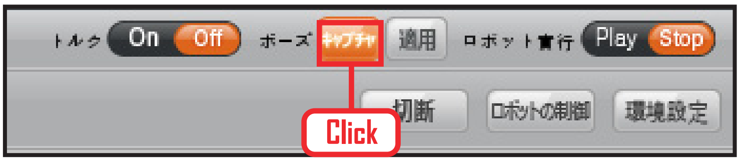

トルク On Off
ポーズ キャプチャ 適用
ロボット実行 Play Stop
切断
ロボットの制御
環境設定
1 Click
2 Click

14　ロボットのキャプチャのためのモーション



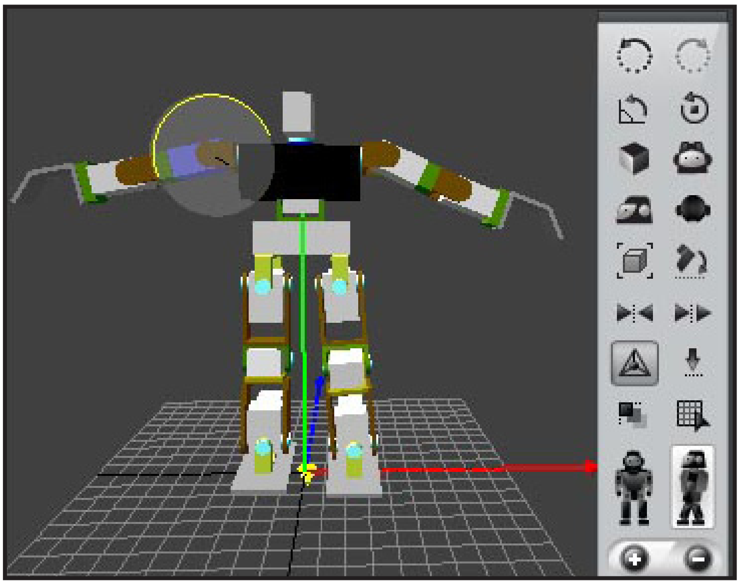
今回はロボット自体をキャプチャして編集する方法について説明します。
“トルクOff”ボタンを押して、ロボットの各関節を自由に動かせる状態でロボットのモーションを手で直接作成します。

15−1　キャプチャ


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの制御
環境設定
Click

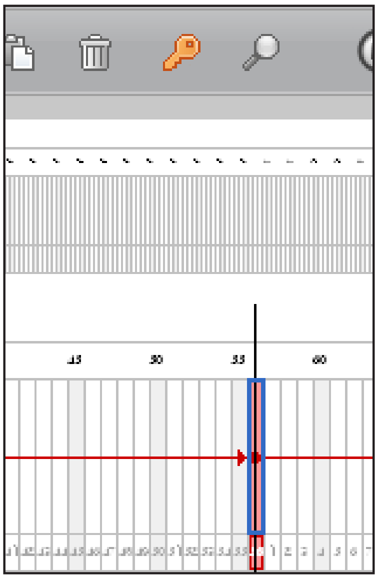
ロボットのキャプチャボタンをクリックします。

15−2　3Dモーションチャンキャプチャ

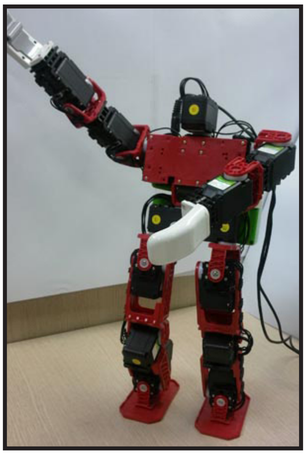


キャプチャボタンを押すと、3Dモーションウィンドウに直接キャプチャしたロボットの形が表示されます。

15−3　キーフレームの挿入

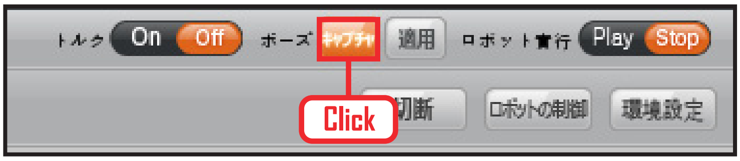


キャプチャしたモーションをフレームに挿入します。
フレームをクリックし、キーフレームをクリックすると挿入されます。



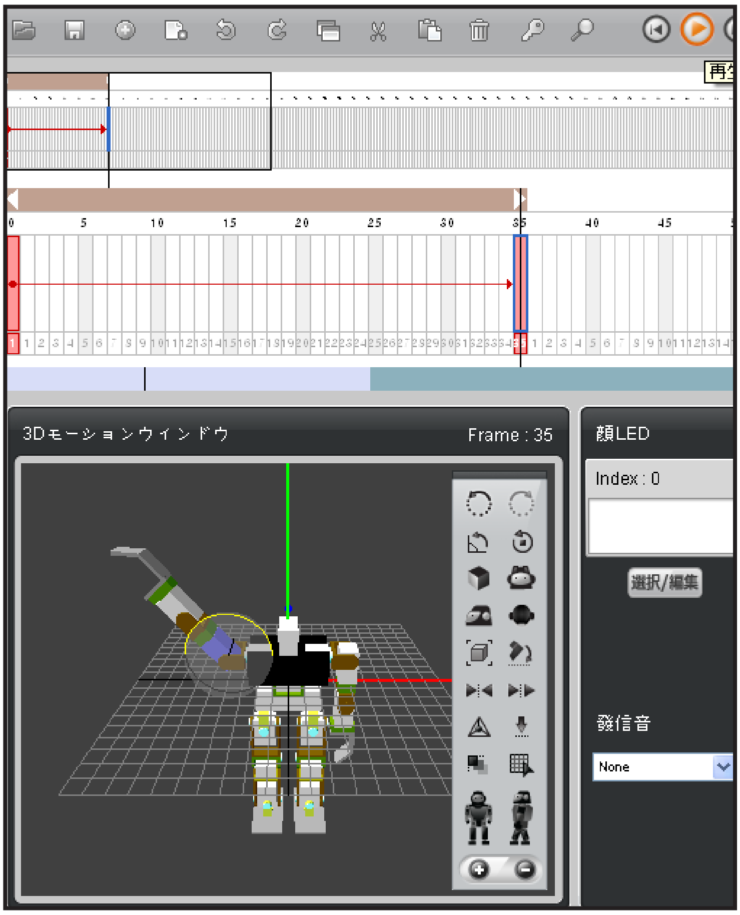
16−1 他の動作

ロボットの他の動作を手で作成します。


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
Click
切断
ロボットの制御
環境設定

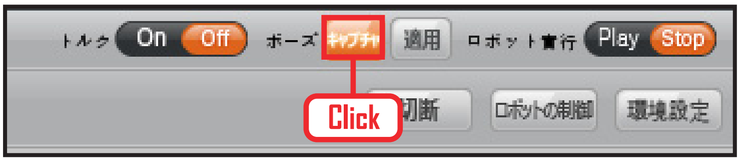
16−2 キャプチャする

キャプチャします。


3Dモーションウィンドウ
Frame : 35
顔LED
Index : 0
選択/編集
發信音
None

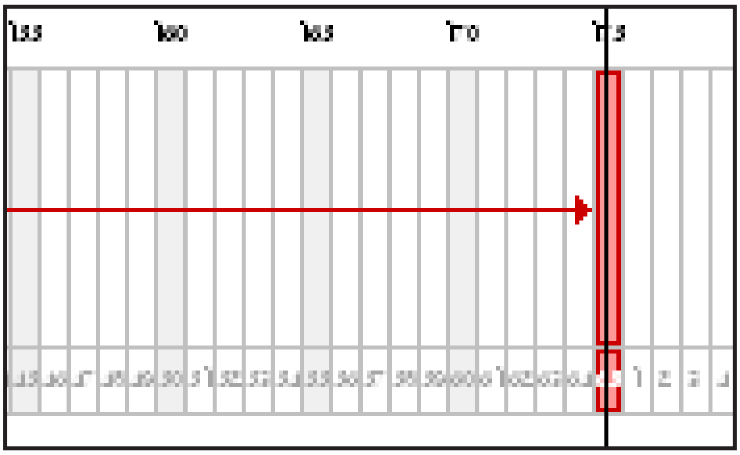
17 モーション確認

手で作成した動作と3Dモーションのウィンドウを比較します。


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
Click
切断
ロボットの制御
環境設定

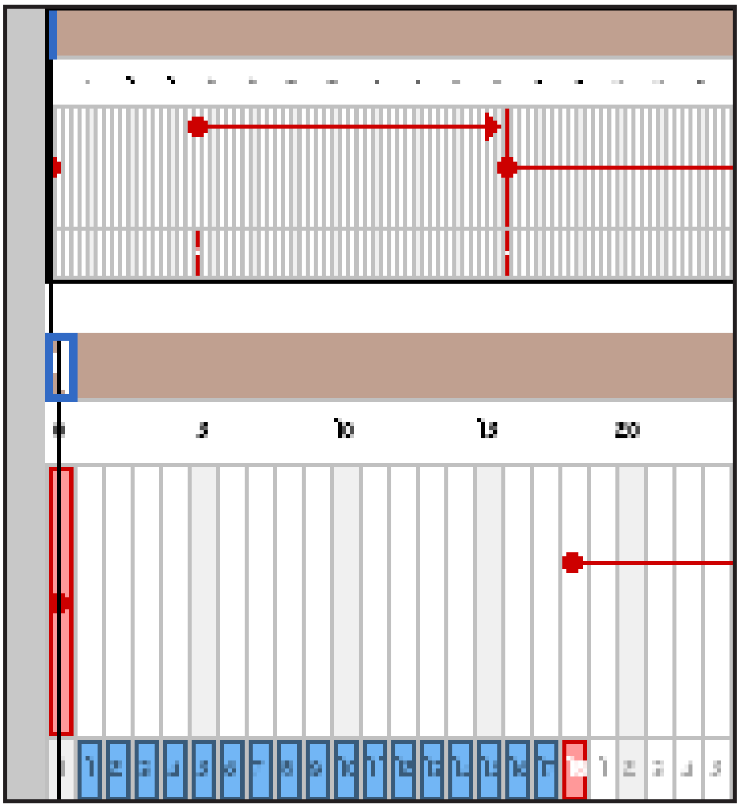
18 キャプチャする

新しく作成したモーションをキャプチャします。


155
160
165
170
175

19 キーフレームの挿入

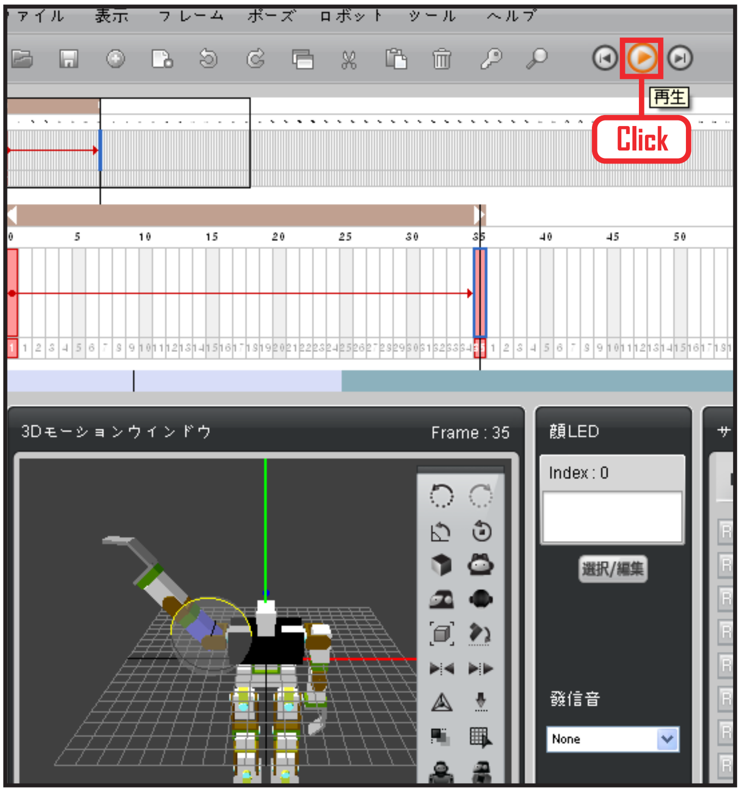
モーションを挿入したい位置に挿入します。



20 Delay値

ロボットの最初の動作では最初の動作のキーフレームと二番目の動作のキーフレーム間にモーションの違いが多くあると、ロボットが急に動きます。それを防止するために最初の動作をゆっくり行うようにする機能を付与します。

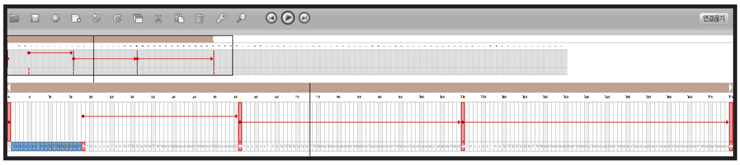
フレームの下部をクリックして、マウスの左ボタンを押したまま右にドラッグすると、左図のように表記されて、Delay値が作成されます。


ァイル
表示
フレーム
ポーズ
ロボット
ツール
ヘルプ
再生
Click
3Dモーションウィンドウ
Frame : 35
顔LED
Index : 0
選択/編集
発信音
None

21　画面再生

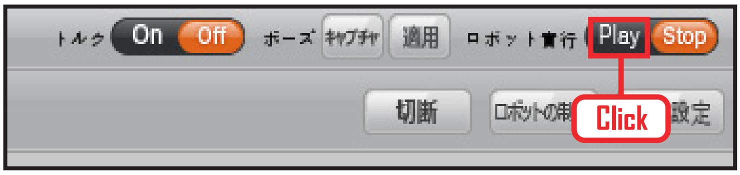
現在までに作成したモーションを画面上で動作できます。

再生アイコンをクリックします。



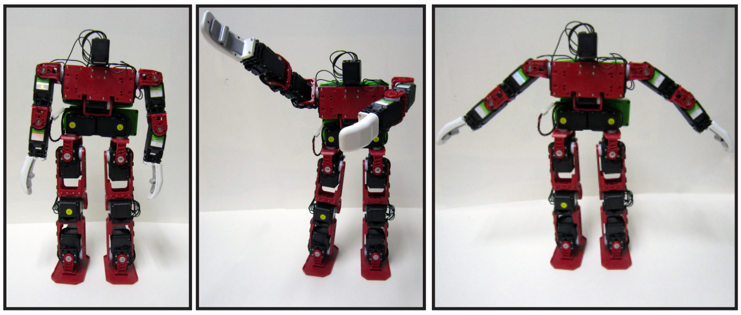
22　再生

タイムフレームで現在再生されている位置に移動線が表記されます。


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの削
Click
設定

23　ロボットの再生

ロボットで直接再生します。右側のロボット接続のアイコンでPlayボタンをクリックします。



24　ロボットの再生

一番左の写真は3Dモーションウィンドウで作成したモーションです。中央の写真はロボットで直接アクションを作成してキャプチャーしたものです。右端の写真もロボットで作成されたモーションをキャプチャしたものです。
再生させると、連続動作で動きます。

ソフトウェア：DR-Visual Logic

インストールする

DR-Visual Logicの紹介

DR-Visual LogicはDongbu Robotで独自に開発したロボットのプログラミング言語をGraphicモジュール化させ、Drag＆Drop方式を採用したロボットプログラミングツールです。Dongbu RobotのDRCコントローラーの機能に合わせて、それぞれの機能をモジュール化しました。マウスのみを使用して、ドラッグ＆ドロップ方式で初心者も簡単にプログラミングすることができる構造になっており、C-likeタブを提供し、グラフィックスプログラミングをTextに変換したコードをすぐに確認することができます。 C文法のようなコードなので、プログラミングの入門者はC文法を学習することもできます。ロボット言語の中で最も簡単で、最も強力な機能を発揮するプログラミングツールとして使われており、初心者から専門家までその活用の幅を広げることによって、ロボット教育の市場で最も注目されています。今後、アップグレードされたコントローラーに関連モジュールを追加し、ロボットのモーションのモジュール化やシミュレーションの組み合わせなど、多彩な機能を発揮するプログラムにアップグレードする予定です。

- システム要件
- 最低インテルペンティアム800Mhz以上
- Windows XP、Windows Vista、Windows7
- 最低256 MB RAM
- ハードディスクの設置スペース300 MBが必要
- USB Port
- Macintosh（対応予定）

PART
06

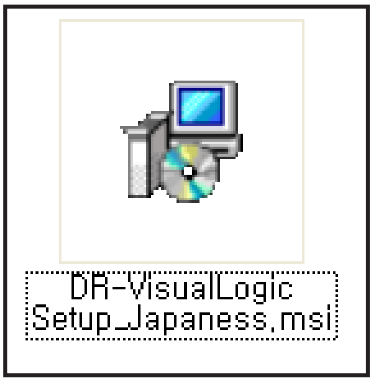
ソフトウェア：DR-Visual Logic

インストールする

インストールから実行まで下の手順で実行してみてください。

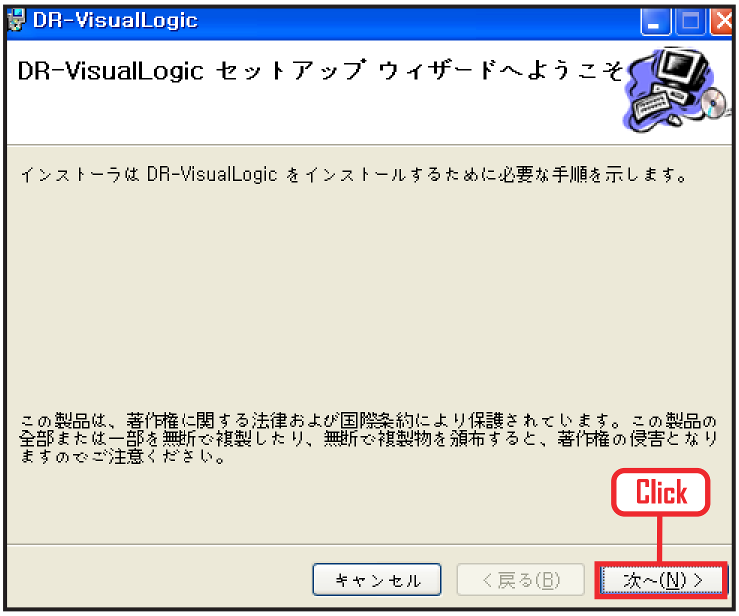
01 インストールファイル

インストールファイルをクリックします。


DR-VisualLogic
Setup_Japaness.msi

02 インストールウィザードの開始

“次へ”ボタンをクリックします。

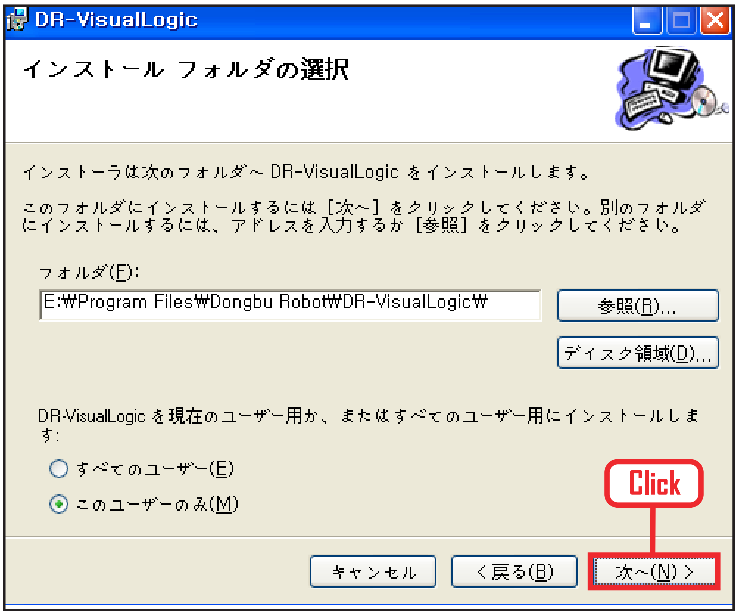


DR-VisualLogic セットアップ ウィザードへようこそ

インストーラは DR-VisualLogic をインストールするために必要な手順を示します。

この製品は、著作権に関する法律および国際条約により保護されています。この製品の全部または一部を無断で複製したり、無断で複製物を頒布すると、著作権の侵害となりますのでご注意ください。

DR-VisualLogic
Click
キャンセル
< 戻る(B)
次へ(N) >


DR-VisualLogic
参照(R)...
ディスク領域(D)...
Click
キャンセル
< 戻る(B)
次へ(N) >

インストール フォルダの選択

インストーラは次のフォルダへ DR-VisualLogic をインストールします。

このフォルダにインストールするには [次へ] をクリックしてください。別のフォルダにインストールするには、アドレスを入力するか [参照] をクリックしてください。

フォルダ(F): E:₩Program Files₩Dongbu Robot₩DR-VisualLogic₩

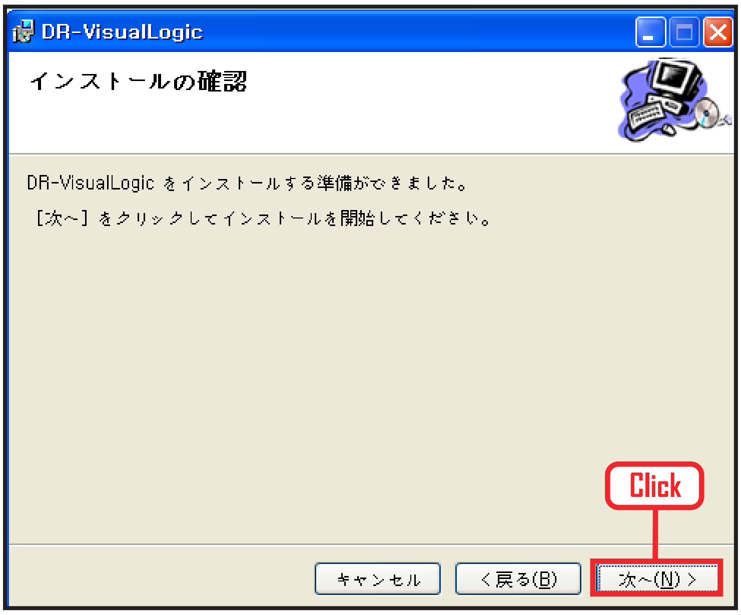
DR-VisualLogic を現在のユーザー用か、またはすべてのユーザー用にインストールします:

○ すべてのユーザー(E)

● このユーザーのみ(M)

03 インストールフォルダを選択

“次へ”ボタンをクリックします。


DR-VisualLogic
Click
キャンセル
< 戻る(B)
次へ(N) >

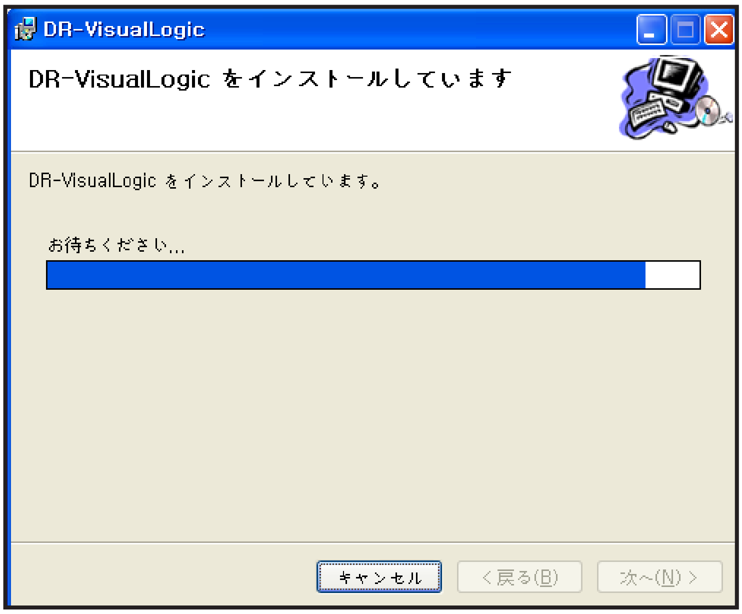
インストールの確認

DR-VisualLogic をインストールする準備ができました。

[次へ] をクリックしてインストールを開始してください。

04 インストール確認

“次へ”ボタンをクリックします。


DR-VisualLogic
キャンセル
< 戻る(B)
次へ(N) >

DR-VisualLogic をインストールしています

DR-VisualLogic をインストールしています。

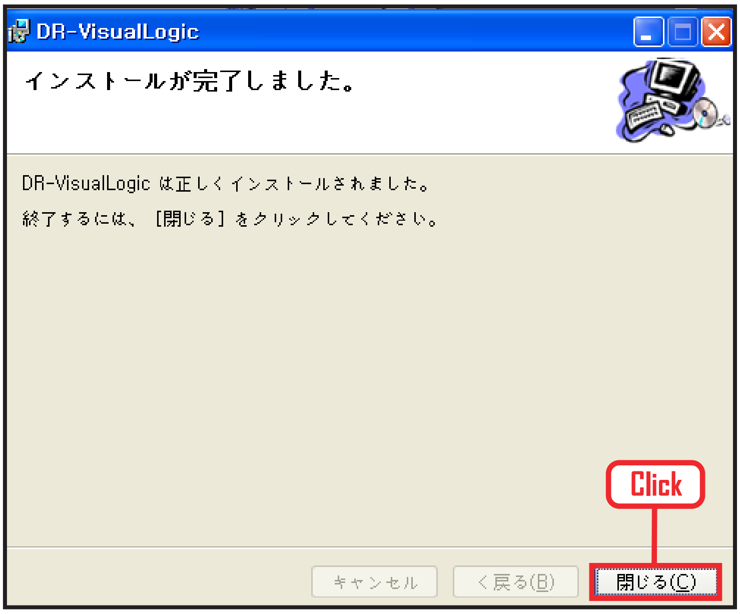
お待ちください...

05 インストールの開始

インストールを開始します。プログレスバーが終わるまでお待ちください

06 インストール完了の選択

“閉じる”をクリックしてください
ソフトウェアのインストールが完了しました。


DR-VisualLogic

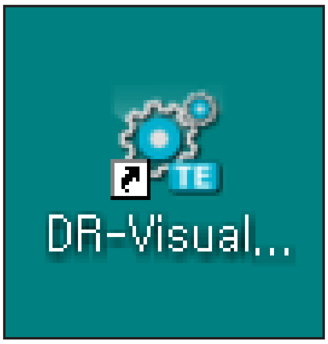
インストールが完了しました。

DR-VisualLogic は正しくインストールされました。
終了するには、[閉じる] をクリックしてください。

Click
キャンセル
< 戻る(B)
閉じる(C)

07 実行ファイルの確認

デスクトップのショートカットアイコンとWindowsのスタート > すべてのプログラム > Dongbu Robot> DR-VisualLogic にある実行ファイルを確認してください。
実行ファイルをクリックすると、プログラムが実行されます。

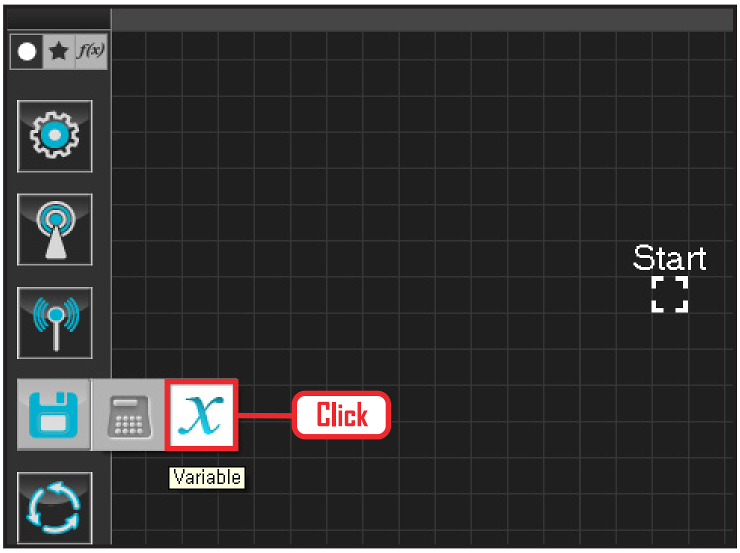

TE
DR-Visual...

Hello DR-Visual Logic

最初のプログラムの実行

例の説明

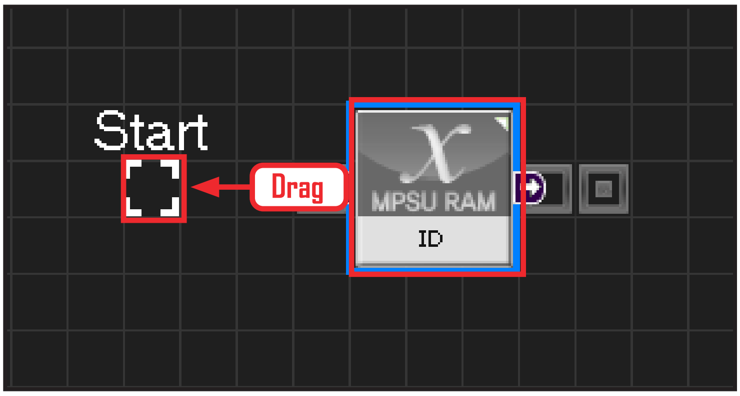
両腕を広げているロボットのモーターを片側だけ下に下げます。 16DOFのヒューマノイドロボットのモーターをすべて中央整列すると、ロボットは両腕を左右に上げる姿勢になります。そのうち、片方の腕を下げます。


f(x)
Start
Click
Variable

01 変数指定

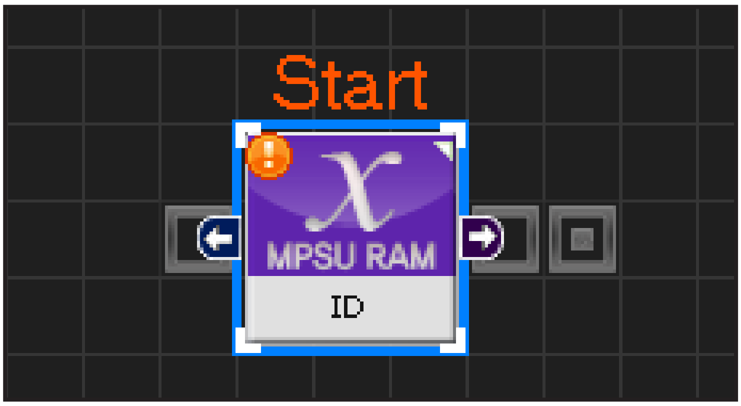
ロボットを動作させることはロボットのサーボモーターを動作させるという意味です。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data> Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

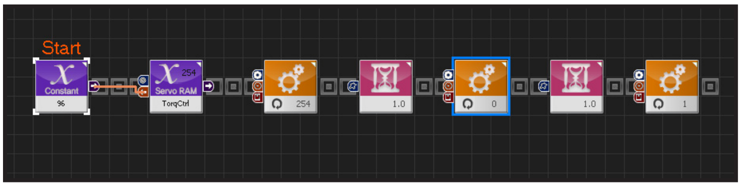
02 スタート

モジュールの左側のコネクタをStart Pointにドラッグして正確にドッキングさせます。


Start
MPSU RAM
ID

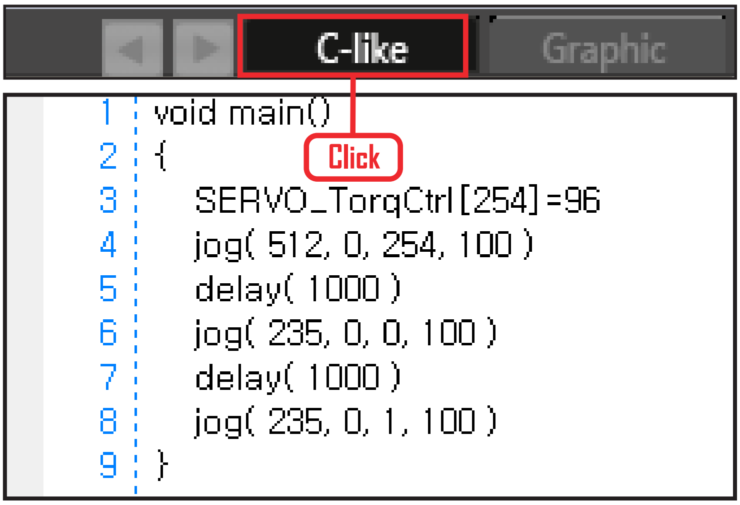
03 プログラミングのスタート

モジュールとStart Pointが正確にドッキングすると、左のように、アクティブなカラーイメージモジュールに変わります。
これはプログラミングが開始されたという意味です。


Start
Constant
96
Servo RAM
254
TorqCtrl
254
1.0
0
1.0
1

04 完全プログラミング

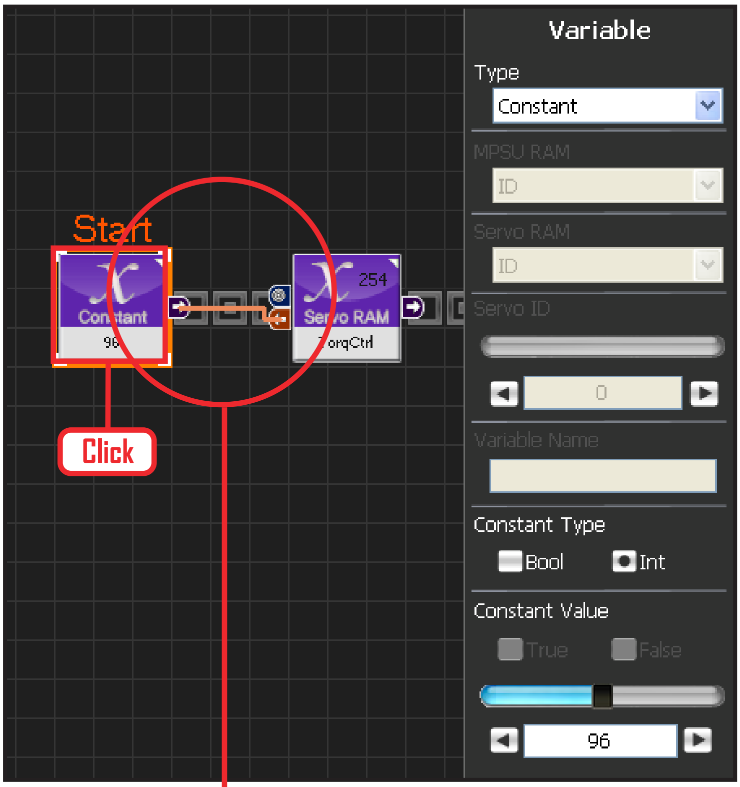
ロボットのモーターを動かして片方の腕を下げる完全プログラミングの展開画面です。


C-like
Graphic
Click

```
void main()
{
   SERVO_TorqCtrl[254]=96
   jog( 512, 0, 254, 100 )
   delay( 1000 )
   jog( 235, 0, 0, 100 )
   delay( 1000 )
   jog( 235, 0, 1, 100 )
}
```

05 C-Like見る

右上のGraphicタブで、C-likeタブをクリックすると、左のようなTaskプログラミングの画面が表示されます。
完全プログラミングTask画面です。Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが動くのでモジュールがTextにどのように変換されているのかを確認することができます。


Start
Constant
96
Servo RAM
254
TorqCtrl
Click

Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

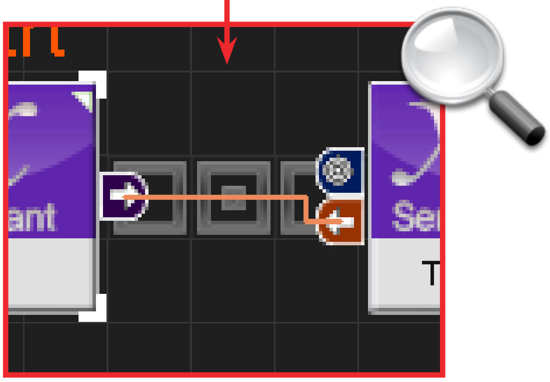
Servo ID: 0

Variable Name:

Constant Type: ☐ Bool ● Int

Constant Value: ☐ True ☐ False

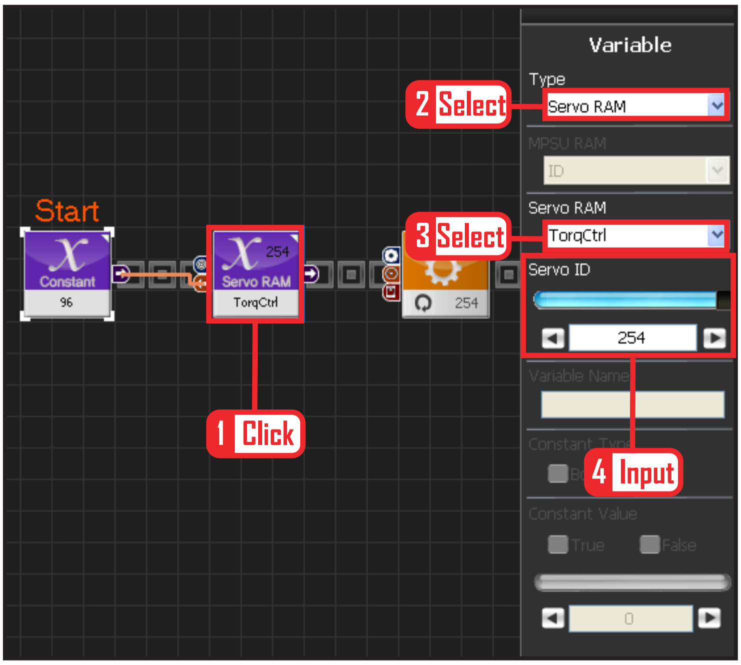
96



06 定数の設定

サーボモーターを自分で動かすことができる状態にする過程です。

Variable TypeをConstantに選択します。
プロパティ中のConstant　Valueの値を96に設定します。
サーボのTorgControlレジスタに96（0X60）という値が入ると、サーボが動くことができる状態になります。
その値はOutputコネクタを介して後のモジュールにトルク値で渡されます。

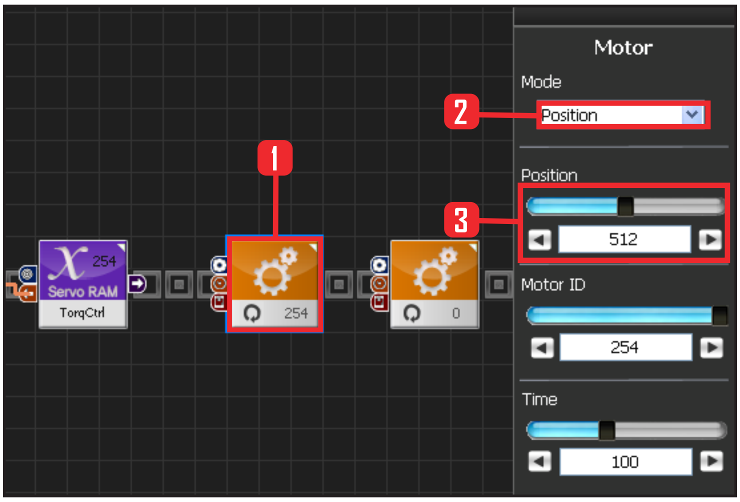

Start
Constant
96
254
Servo RAM
TorqCtrl
254
1 Click
2 Select
3 Select
4 Input
MPSU RAM
ID
Variable Name
Constant Type
Constant Value
True
False
0

Variable
Type: Servo RAM
Servo RAM: TorqCtrl
Servo ID: 254

07 全サーボに適用

前にもらった96定数値を全サーボに適用する過程です。

Variable > Type：Servo RAMを選択します。
Servo RAM：TorqCtrlを選択します。
Servo　ID：254に設定します。 254は接続されている全サーボに適用するという意味です。

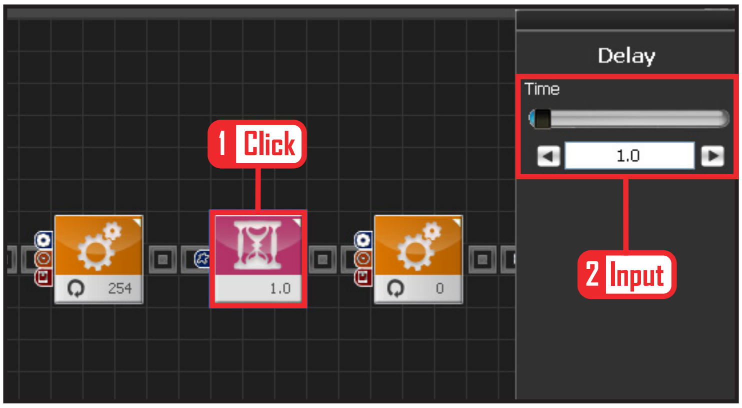

254
Servo RAM
TorqCtrl
254
0
1
2
3

Motor
Mode: Position
Position: 512
Motor ID: 254
Time: 100

08 全サーボモーターの角度を設定

全サーボモーターの角度を中央に位置させる過程です。

Motion > Motorを選択します。
Mode：Positonに選択します。角度を調整します。
Position：512に設定します。　512番はモーターを中央に移動させるという意味です
Motor　ID：254に設定します。 254はすべてのモーターに適用するという意味です。
Time：100に設定します。単位は1あたり11.2msで、100は約1.12秒を意味します。
1.12秒の間に設定した角度に位置させるという意味です。


254
1.0
0
1 Click
2 Input

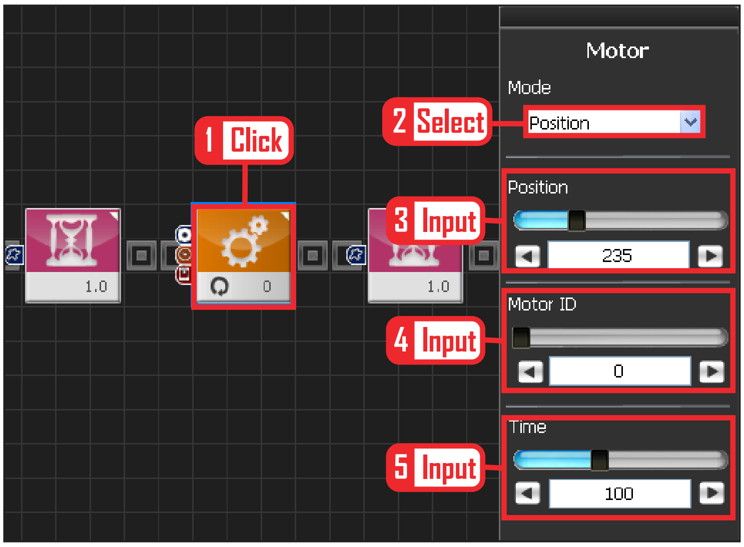
Delay
Time: 1.0

09 Delay

次のモーターの動作前に1秒待ってから起動させる設定です。

Flow > Delayモジュールを選択します。
Time：1.0を設定します。　1秒ぐらいDelayするという意味です。

10 モーター0番(右肩)の設定

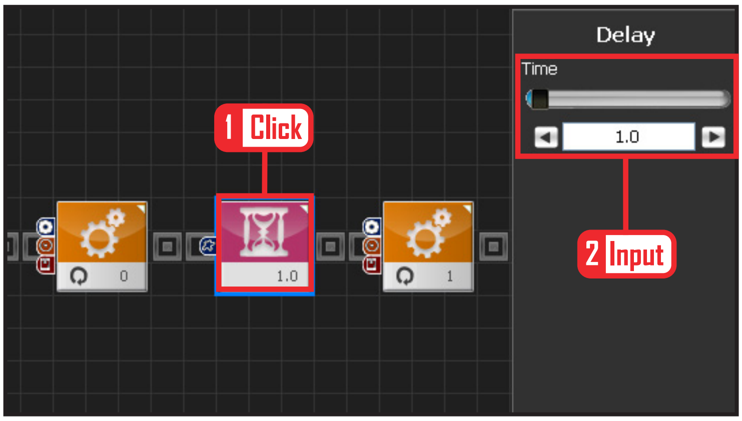

1 Click
2 Select
3 Input
4 Input
5 Input
1.0
0
1.0

Motor
Mode: Position
Position: 235
Motor ID: 0
Time: 100

基本姿勢を作成する
すべてのロボットのモーターの角度を中央に整列させると、ヒューマノイドは腕を左右に広げます。このとき片腕を基本姿勢に変更する設定です。

Motion > Motor を選択します。
Mode:Position を選択します。
Position:235に設定します。 235は水平になっている右手を垂直に下げるためにモーターを回します。
Motor ID:0に設定します。右肩モーターのIDが0番です。
Time:100に設定します。約1.12秒間設定角度に移動します。

11 Delay

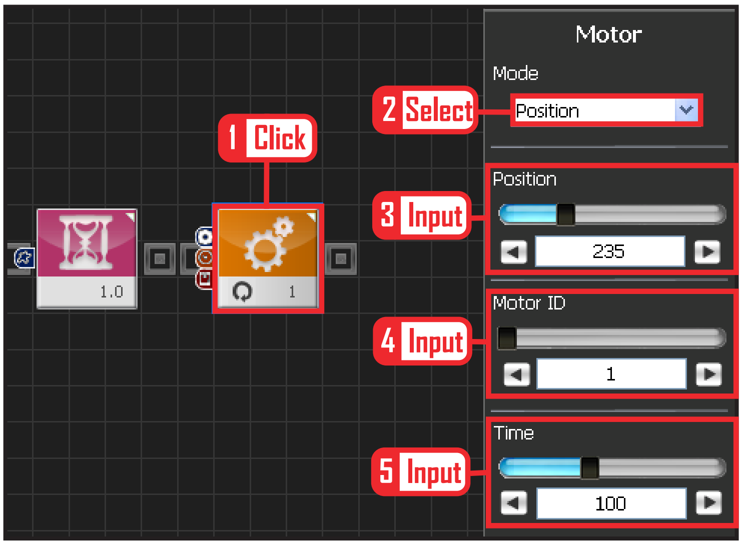

1 Click
2 Input
0
1.0
1

Delay
Time: 1.0

次のモーターの動作前に1秒待ってから起動する設定です。

Flow>Delayモジュールを選択します。
Time:1.0を設定します。1秒ぐらいDelayするという意味です。

12 モータ1番(右腕)の設定


1 Click
2 Select
3 Input
4 Input
5 Input
1.0
1

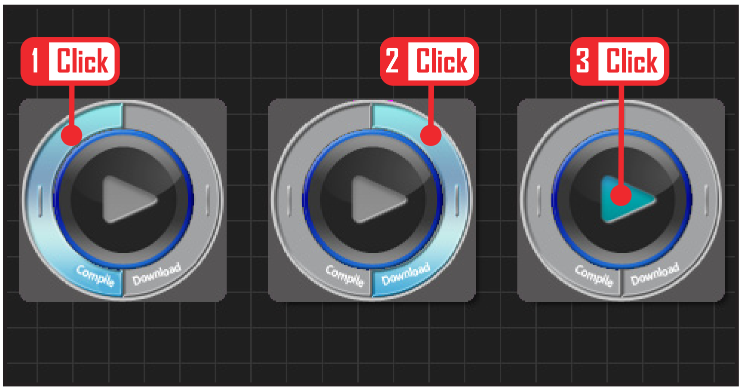
Motor
Mode: Position
Position: 235
Motor ID: 1
Time: 100

Mode:Postionを選択します。
Position:235を選択します。 235は水平になっていた腕を垂直に下げます。
Motor ID:1に設定します。右上の腕の肩に接続しているモーターが1番のモーターです。
Time:100に設定します。約1.12秒間設定角度に移動します。

13 ダウンロード

プログラミング後、コンパイル-> ロボットにダウンロード -> 実行する処理を行います。

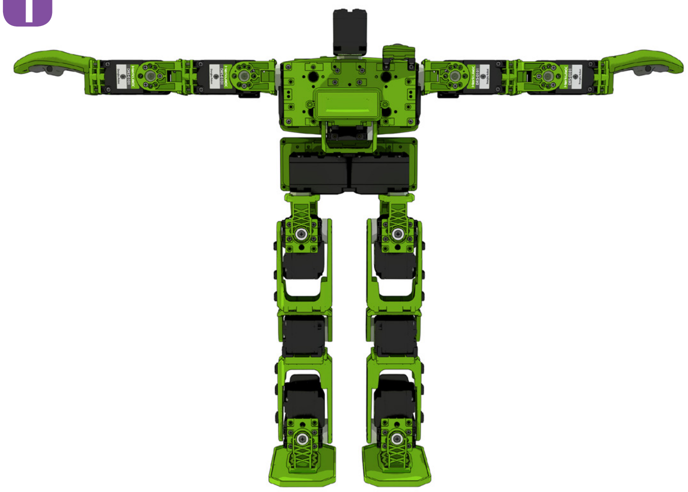
Compileをクリックします。エラーがなければ、右のDownloadをクリックします。ロボットにダウンロードします。ダウンロード完了後、[実行]ボタン（中央の矢印）をクリックします。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

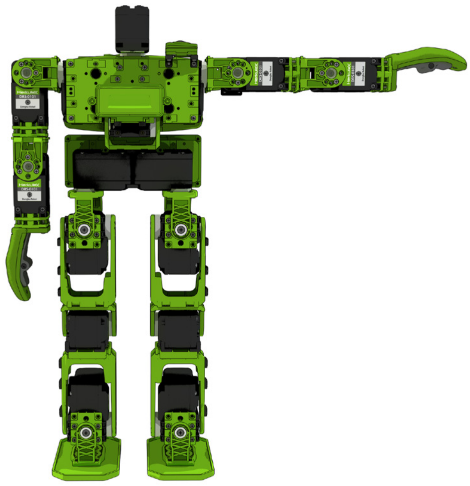
14 ロボットの動作

両腕を広げて水平を維持していた腕の中の右腕が下に動きます。

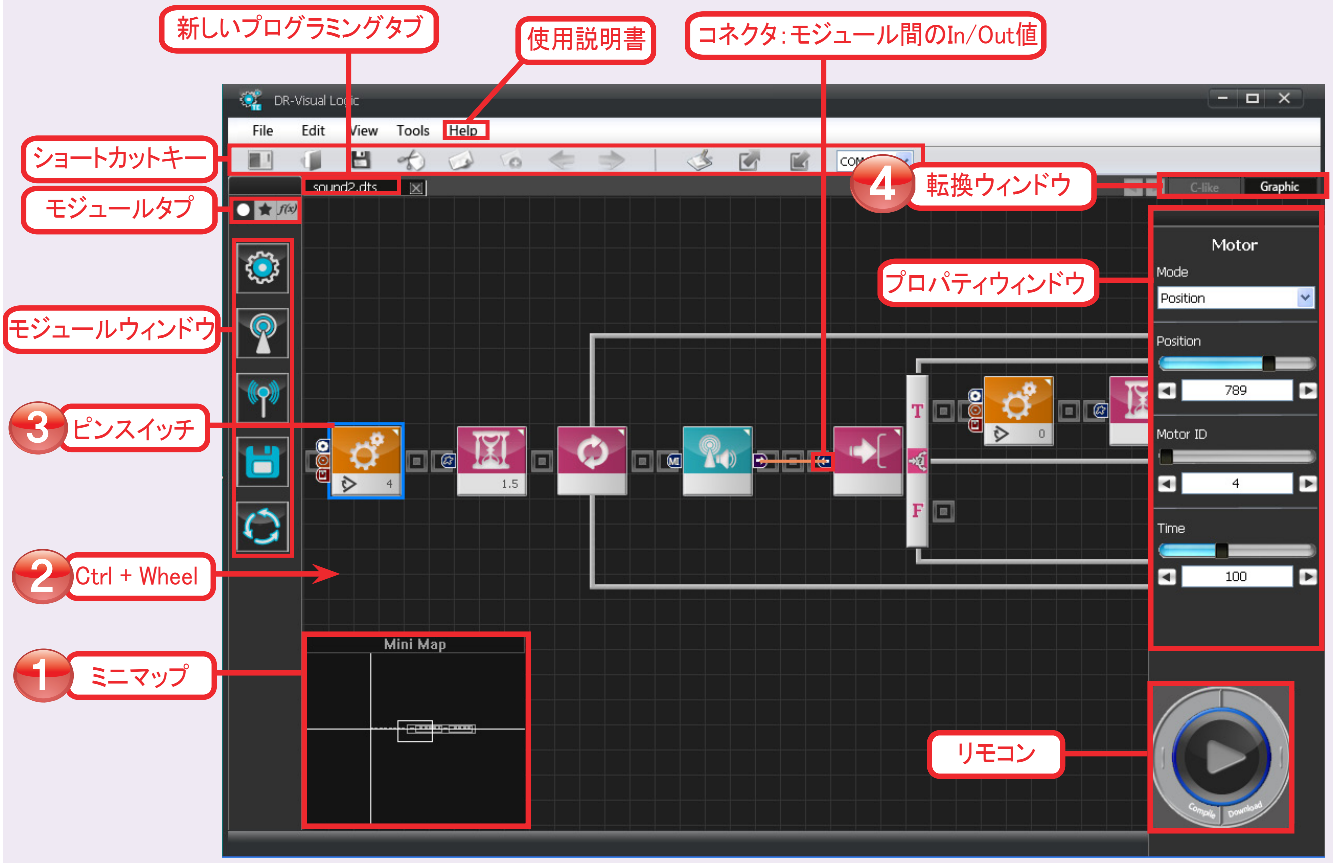
1



2



ユーザーインターフェース


新しいプログラミングタブ
使用説明書
コネクタ：モジュール間のIn/Out値
ショートカットキー
モジュールタブ
モジュールウィンドウ
③ ピンスイッチ
② Ctrl + Wheel
① ミニマップ
④ 転換ウィンドウ
プロパティウィンドウ
リモコン
DR-Visual Logic
File Edit View Tools Help
sound2.dts
C-like Graphic
Motor
Mode
Position
Position
789
Motor ID
4
Time
100
Mini Map
T
F
Compile Download

① ミニマップ：ドラッグ方法で調整すると、プログラムが長くなっても、現在の位置を知ることができ、任意の場所に一気に移動できます。

② Wheel Up/ Down：画面の拡大縮小

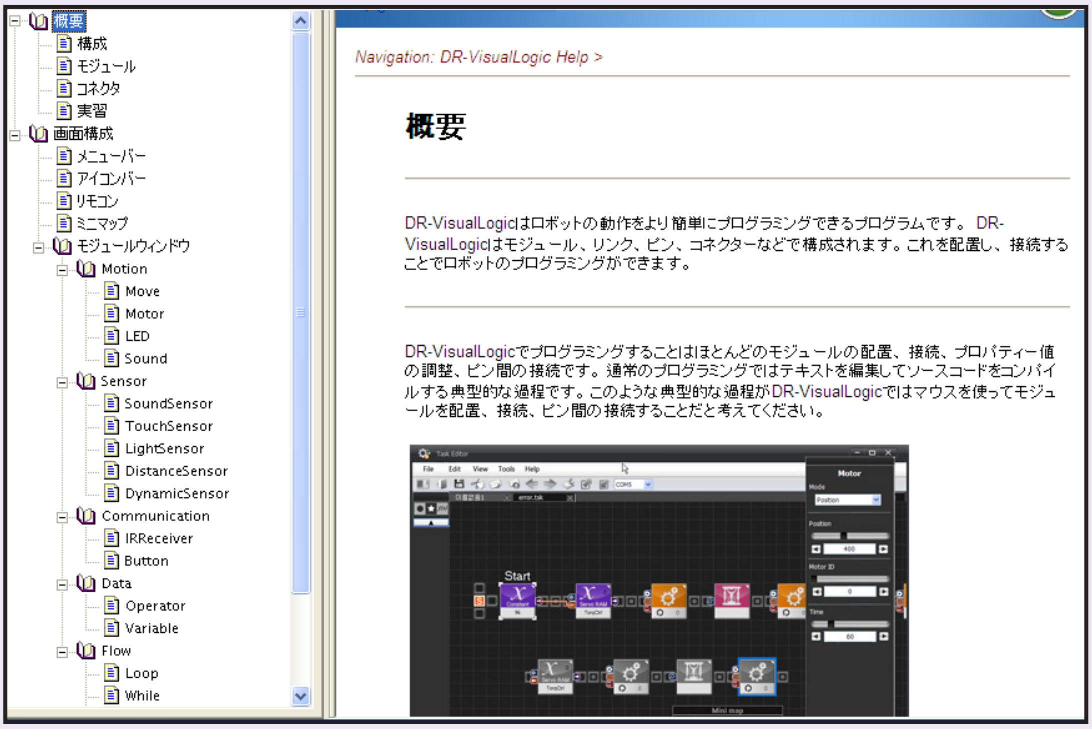
③ ピンスイッチ：クリックすると、モジュールのピンのタイトルが出て一目で分かるように処理されます。もう一度押すと、消えます。

④ 転換ウィンドウ：Graphic to Textグラフィックスでプログラムされたソースが Text形式のソースに変換されます。C文法と似ています。

⓪ ショートカットキー：メニューの中でよく使われるメニューをショートカットキーとして処理
① モジュールタブ：すべてのモジュール
② モジュールウィンドウ
③ 新しいプログラミングタブ
④ 使用説明書
⑤ コネクタ
⑥ プロパティウィンドウ
⑦ ダウンローダ

ソフトウェア：DR-Visual Logic

メニューの ツール＞ヘルプ をクリックすると、以下のようなヘルプのポップアップウィンドウが表示されます。DR-Visual Logicを使用するためには先ずヘルプをお読みいただいた上で使用することをお勧めします。（上部 ［ヘルプ］をクリック ＞ ［目次］をクリック ＞ 左側のメニューの[タイムライン]をクリック ➜ 下の図が表示されます）


概要
構成
モジュール
コネクタ
実習
画面構成
メニューバー
アイコンバー
リモコン
ミニマップ
モジュールウィンドウ
Motion
Move
Motor
LED
Sound
Sensor
SoundSensor
TouchSensor
LightSensor
DistanceSensor
DynamicSensor
Communication
IRReceiver
Button
Data
Operator
Variable
Flow
Loop
While
Task Editor
File Edit View Tools Help
Start
Motor
Mode
Position

Navigation: DR-VisualLogic Help >

概要

DR-VisualLogicはロボットの動作をより簡単にプログラミングできるプログラムです。 DR-VisualLogicはモジュール、リンク、ピン、コネクターなどで構成されます。これを配置し、接続することでロボットのプログラミングができます。

DR-VisualLogicでプログラミングすることはほとんどのモジュールの配置、接続、プロパティー値の調整、ピン間の接続です。通常のプログラミングではテキストを編集してソースコードをコンパイルする典型的な過程です。このような典型的な過程がDR-VisualLogicではマウスを使ってモジュールを配置、接続、ピン間の接続することだと考えてください。

- 概要
  - 構成
  - モジュール
  - コネクタ
  - 実習
- 画面構成
  - メニューバー
  - アイコンバー
  - リモコン
  - ミニマップ
  - モジュールウィンドウ

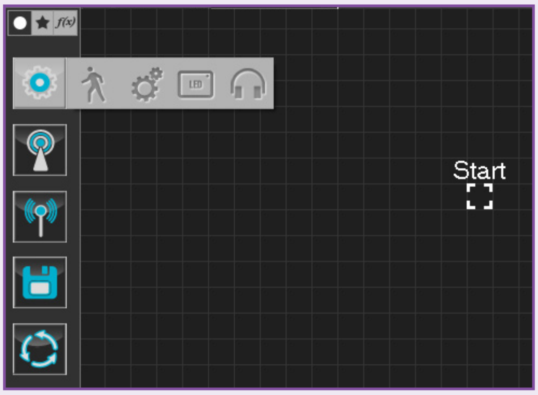
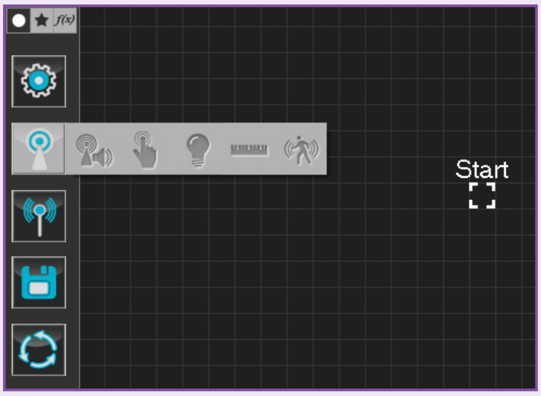
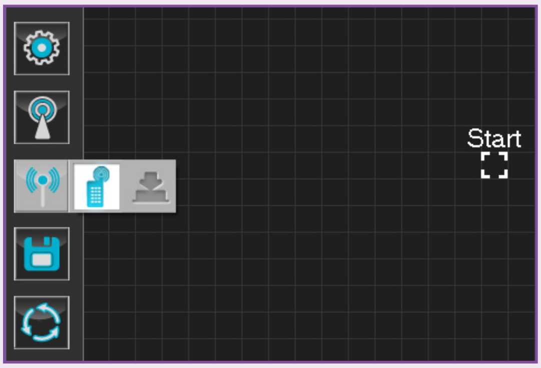
- モジュールウィンドウ

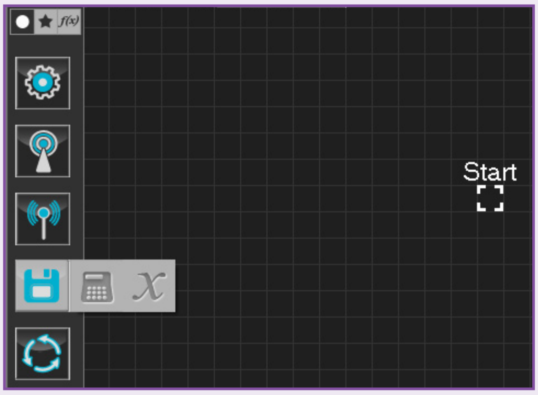
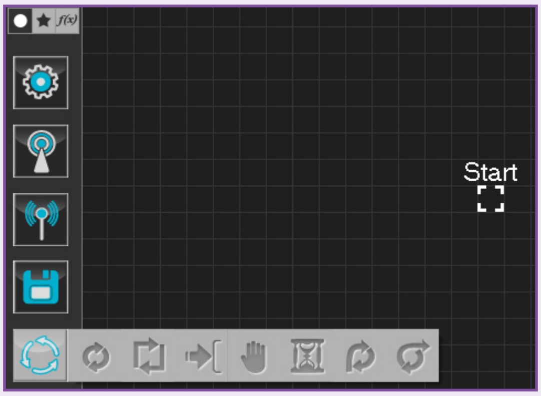
Motio： Move(ロボット保存モーション), Motor(サーボモーター),
LED, Sound(ビープ音)
Sensor： Sound Sensor, Touch Sensor, Light Sensor,
Distance Sensor(距離センサ, PSD Digital, PSD Analog),
Dynamics Sensor(Accermeter, Gyro センサ)
Communication ：IRReceive, ZigBee, Button
Data : Operator, Variable
Flow : Loop, While, Switch, Wait, Delay, Continue, Break

プログラミングのモジュール

DR-Visual Logicは以下のモジュールで構成されています。

Module Packにはプログラムを作るために必要なすべてのプログラミングのモジュールが含まれています。それぞれのモジュールはDRCコントローラーを支援する機能があります。Descriptionはヒューマノイドの組み立て時の各部品の位置と、その部分に該当するモジュールについて簡単に説明しています。

Module Pack	Picture	Module	Description
Motion （モーション）		Move	保存されたロボットのモーションを実行する。
		Motor	モーター別の位置/速度制御
		LED	頭 – 保存されたLEDを実行する。 背中 – コントローラーのLEDを制御する。
		Sound	Buzzerの音量
Sensor （センサ）		Sound Sensor	左右のサウンドを区別
		Touch	頭のモジュールのタッチの有無を認識
		Light	背中部分の光量を測定
		Distance	距離測定
		Dynamics	加速度と角速度を測定
Communication （通信）		IRRciever	リモコンデータの認識
		Button	背面のコントローラーのボタンを認識

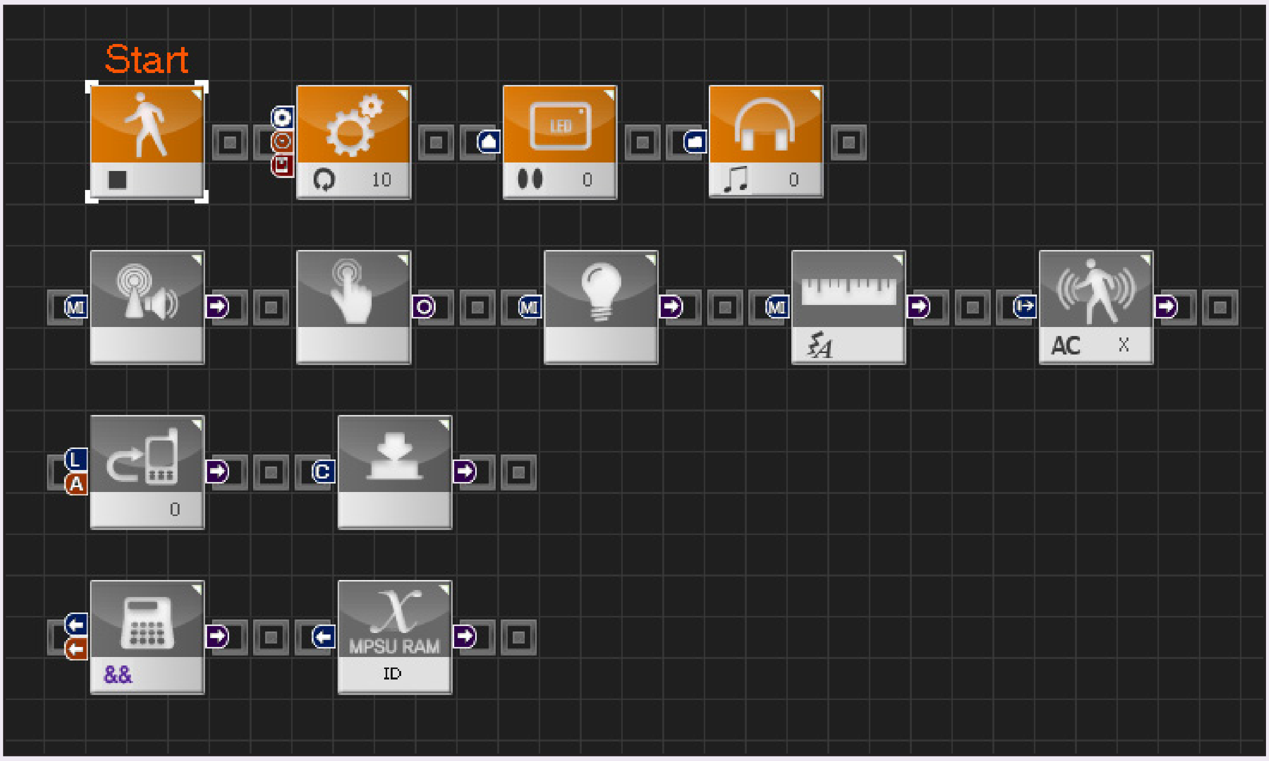
Module Pack	Picture	Module	Description
Data （データ）		Operator	演算子
		Variable	レジスタデータのユーザー定義の変数/定數
Flow （流れ分岐）		Loop	無限反復/ for文
		While	特定の条件の反復
		Switch	管理分岐、if-else
		Wait	特定条件の待機
		Delay	指定時間の待機
		Continue	loopの最初へ戻る
		Break	loopを抜ける

プログラミングモジュール>一般型モジュール

DR-Visual Logicのモジュールは一般型モジュールとFlow型のモジュールに分かれています。

一般型モジュール

一般型モジュールは連続して、モジュール同士を接続して使用されます。Flowモジュールを除くすべてのモジュールがこれに該当します。

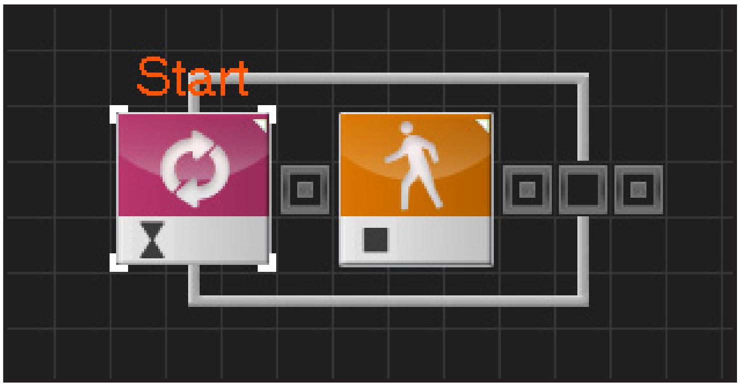

Start
10
LED
0
0
AC
X
0
&&
MPSU RAM
ID

上から順番にMotion, Sensor, Communicaiton, Dataを示すモジュールのアイコンです。

プログラミングモジュール> Flow型モジュール

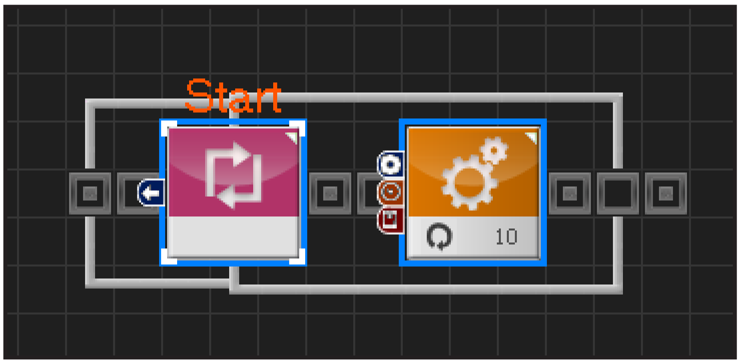
Flow型モジュール

Flow型モジュールは通常のモジュールと接続して、繰り返し、分岐などのプログラミングの流れの中に接続できるモジュールです。このため、一般型のモジュールとは異なり、通常のモジュールとの接続時、アウトラインの形の帯が形成されるグラフィックスの構造を持っています。



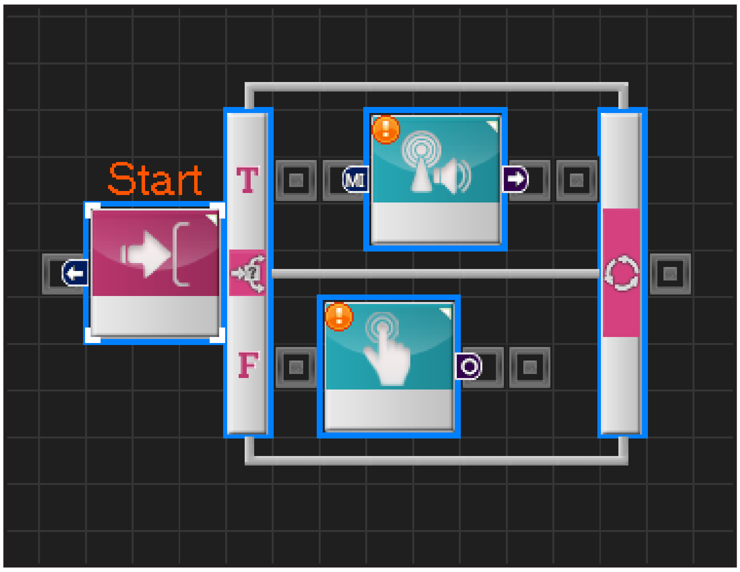
Loop

Loopは繰り返しを指示するモジュールとして、一定の回数だけ繰り返すFor文と永遠に繰り返すForever文があります。



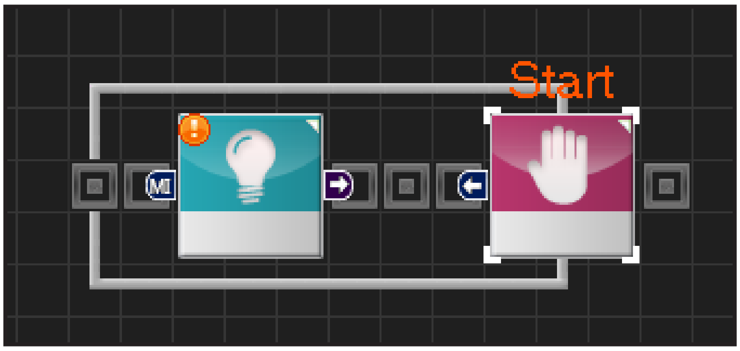
While

Whileは前の条件を満足する時に後ろを実行するという意味で、条件と結合された繰り返し文です。



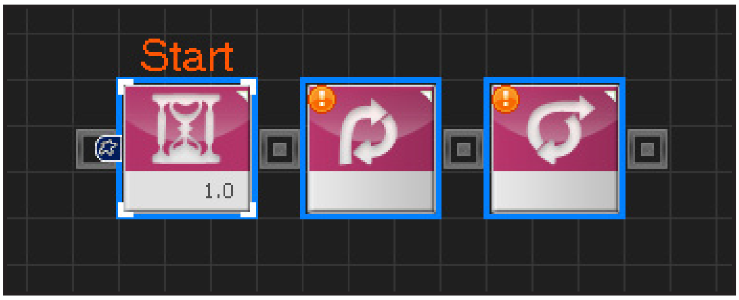
Switch

入力条件件が真の場合、上を、偽の場合、下を実行するif-else文です。



Wait

入力条件が真の間、プログラムの実行を停止し、偽の場合、実行を再開します。



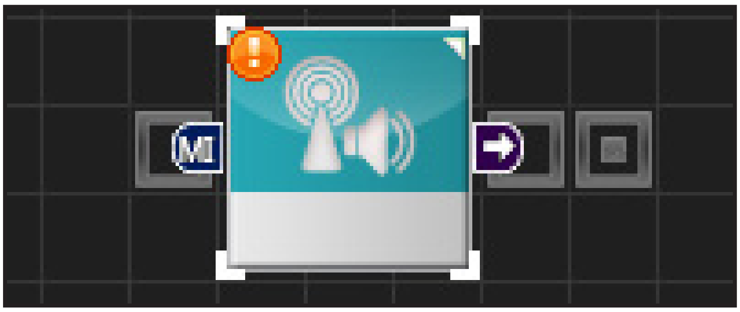
Delay, Continue, Break

左の3つのモジュールは周囲に帯が形成されない一般形のモジュールのように表示されます。
Delayは一定時間遅延させるモジュールです。
Continueは繰り返し文の最初に戻って継続して実行するという意味です。
Breakはループを抜け出すという意味です。

プログラミングモジュール>コネクタ

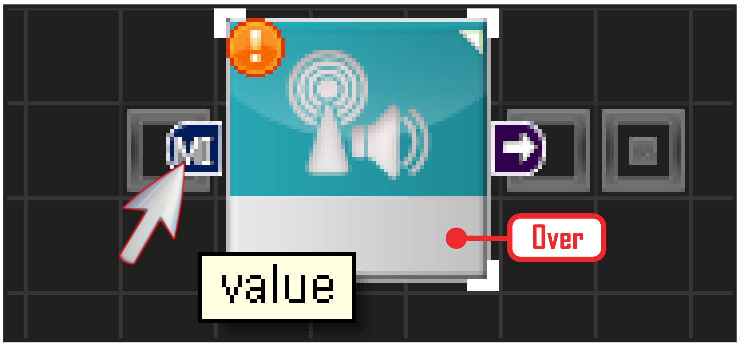
コネクタ

モジュールによってInput値やOutput値が持つモジュールがあります。　Outputコネクタの結果値はこれに接続しているモジュールのInput値に入力して動作します。 Input/Output値を持つモジュールはモジュール上に最初から左右の側面にコネクタの形で示されています。以下はコネクタの例を示します。



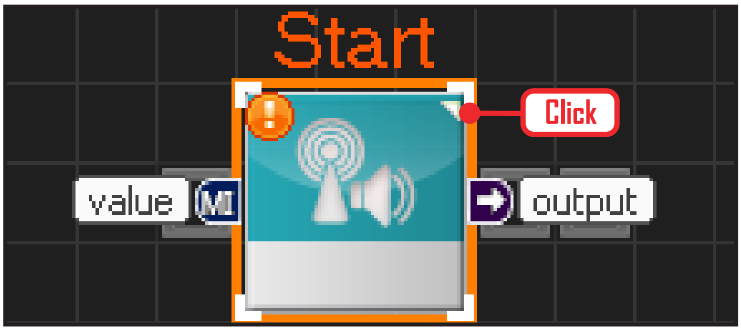
コネクタ

腕を上げられたモーションをフレームに挿入します。目的のフレームをクリックし、キーのアイコンをクリックして、モーションを挿入します。



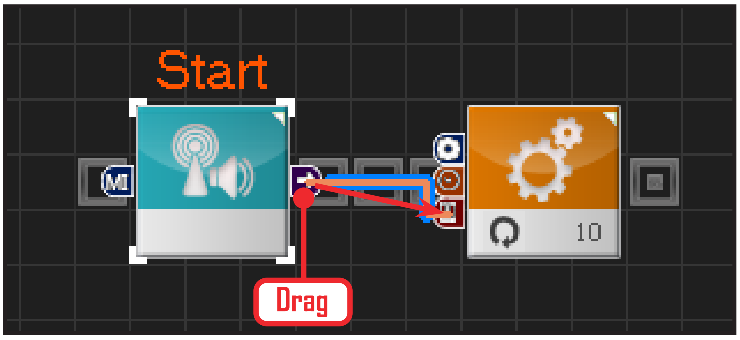
ふきだしヘルプ

コネクタのアイコンのみを見てどの機能のコネクタなのか区分することは困難です。コネクタの上にマウスを置くと左のようにふきだしでピンの名前が表示されます。



ふきだし展開

複数のコネクタの名前を一度に見たい場合、右上の三角の形のアイコンをクリックすると、コネクタの両端にコネクタの名前のふきだしが表示されます。もう一度クリックすると消えます。



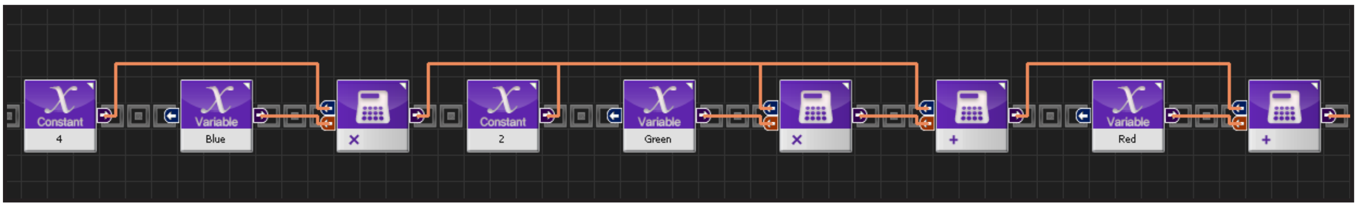
コネクタの接続

前のモジュールのOutput値を後ろのモジュールのInput値に入れるためにマウスでドラッグして接続すると、左のように接続線が表記されます。

プログラミングモジュール>連結型タイプ

連結型タイプ

モジュールを接続する時、直列接続と並列接続があります。

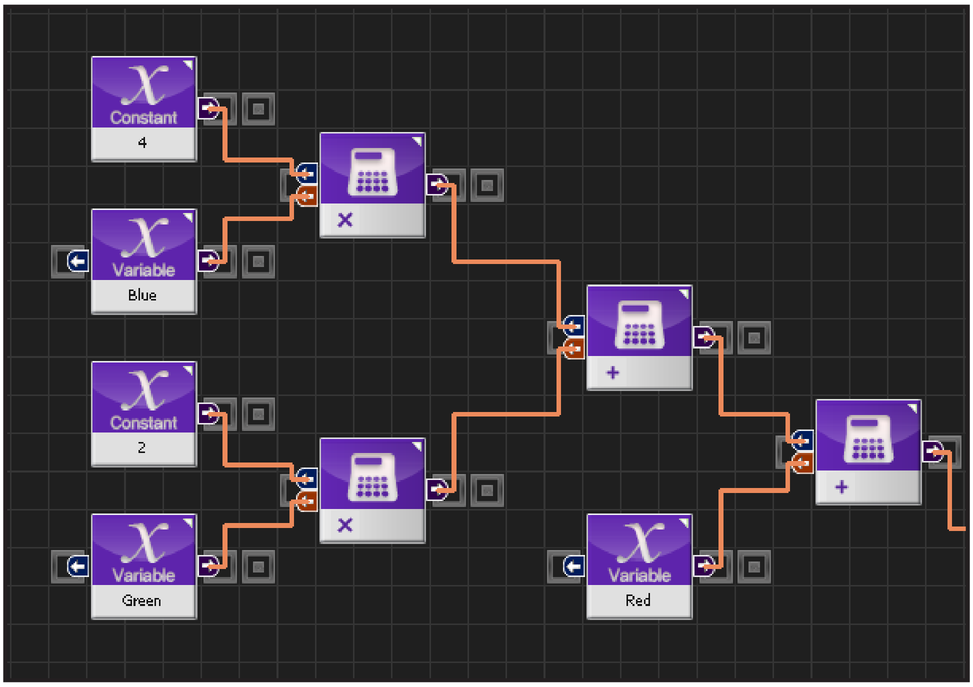

Constant
4
Variable
Blue
×
Constant
2
Variable
Green
×
+
Variable
Red
+

直列型連結

直列型接続は文字通り、左から右へのモジュールを順次接続することです。

上記に演算に関するプログラミングをしめしています。

((4xBlue)+(2xGreen))+(1xRed)の演算式を直列型連結で表したものです。


Constant
4
Variable
Blue
Constant
2
Variable
Green
×
×
+
Variable
Red
+

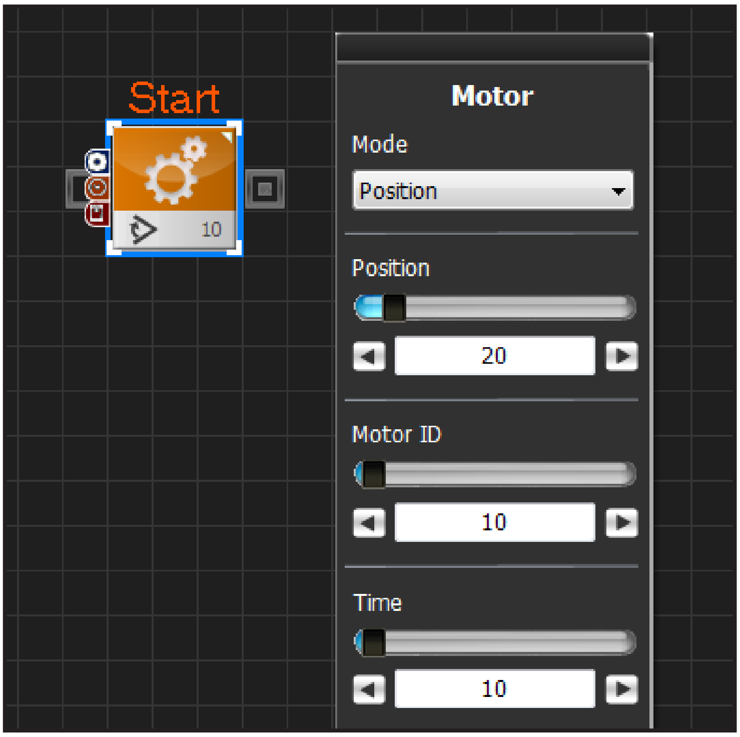
並列型連結

並列型連結は上下の空間を利用して、モジュールを並列に接続することです。次の例は上記の直列型連結と同じ演算のプログラミングです。

プロパティウィンドウ

プロパティウィンドウ

モジュールごとに独自のプロパティを持っていて、そのプロパティ値を設定すればプログラミングを行うことができます。リストのポップアップ、ラジオボタン、数値の設定などのUIで表現されています。各モジュール別プロパティの説明とプロパティ値/制限値などはヘルプを参照してください。



Motor

Mode: Position

Position: 20

Motor ID: 10

Time: 10

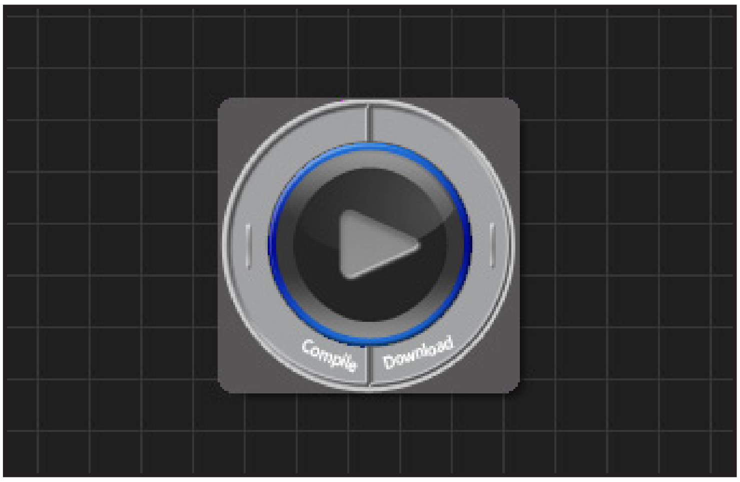
プロパティウィンドウ

モーターモジュールをクリックした時に右側のようなプロパティウィンドウが表示されます。
モーターは位置制御と速度制御のプロパティがあります。　位置制御のためにはModeでPostionを選択し、速度制御のためにはVelocityを選択します。
詳細設定でPosition、Motor　ID、Timeなどの値を調整します。

コンパイル/ダウンロードする

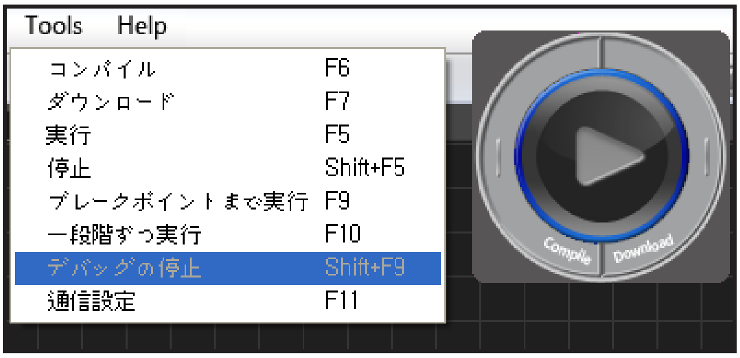
ダウンローダ

ロボットのプログラミングを完了し、コンパイル、ダウンロード、ロボットの実行を行います。プログラムウィンドウの左下に大きなアイコンで提供しており、ツールメニューから活用できます。


Compile
Download

ダウンローダのアイコン

左のアイコンはダウンローダのアイコンです。左側はCompile、右側はDownloadを表し、中の矢印はロボットでの実行を意味します。


Tools
Help
コンパイル F6
ダウンロード F7
実行 F5
停止 Shift+F5
ブレークポイントまで実行 F9
一段階ずつ実行 F10
デバッグの停止 Shift+F9
通信設定 F11
Compile
Download

ツールメニュー

ツールメニューではより詳細な実行が可能です。

コンパイル：作成したTaskをコンパイルします。
ダウンロード：コンパイルしたTaskをダウンロードします。
実行：ダウンロードしたTaskを実行します。
停止：ロボットの実行中に停止します。
ブレークポイントまで実行：プログラム中間ごとにブレークポイントを指定すると、そこまで実行後、一時停止します。
一段階ずつ実行：モジュールごとに実行します。

モジュール別のプログラミング

基本提供しているサンプルプログラムはDRCコントローラープラットフォーム基盤のプログラムであり、基本的に16DOFのヒューマノイドに基づいて作成されています。　18、20DOFのヒューマノイドや変身組立型のロボットでのモーションやモジュールを変更する時はそれに合わせて再びプログラムする必要があります。
プログラムの適用時にはまずロボットのモーターのIDを確認して、ロボットのセンサの位置を確認してください。また、DR-SIMを通してロボットに保存したモーションのリストをチェックして、プログラミング時に合うIndex値を与えます。基本提供するサンプル例は以下の通りです。

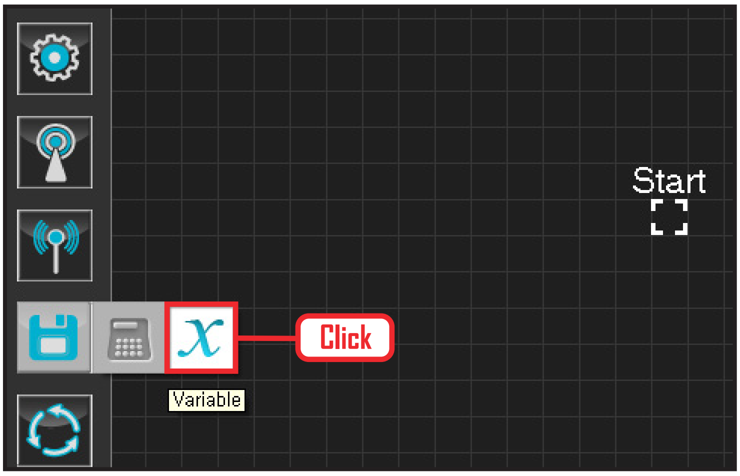
Module Pack	Module	サンプルの例
Motion（モーション）	Move	MoveはDRCコントローラーに保存されたロボットのモーションを取得し、プログラムに適用します。ロボットのモーションを番号でインポートします。番号ごとの動きの名前はDR-SIMで確認することができます。DR-SIMで編集された特定のモーションをロボットが無限に繰り返すプログラミングをします。ロボットの信頼性テストや展示などで活用度が高く、難易度が高いプログラムなので、一つずつ実行してください。
	Motor	モーターを個別に制御してダンスをするプログラムです。
	LED	DRCコントローラーのボタンを押すと、LEDをオン/オフさせるプログラムです。（With Button）
	Sound	リモコンのボタン入力（1番～8番）を受けて音を鳴らすプログラムです。（With IRRecieve）
Sensor（センサ）	Sound Sensor	Sound SensorはDRCコントローラー内部の両側にあります。左側で拍手をすると、左手を上げます。右側で拍手をすると、右手を上げるプログラミングです。（Sound Sensor Sample 1番） 周囲の騒音が多い時は両側からの音をそれぞれ区別することが難しくなります。片側で拍手をしても両手を上げたり、不規則に反応します。周囲の騒音があっても、ロボットが正確な反応をするために、より細分化されたプログラミングをする必要があります。最初に音が入力された時に他の音が入力されないように、強制的にDelayを与えることによって、一度拍手をした時に一度だけ腕を上げるように精度を向上させるプログラムです。（Sound Sensor Sample 2番）
	Light	光の強さに応じてロボットのモーターを動作させる例です。外部の光が暗くなると、ロボットが左腕を上げます。（コントローラーの背面にあるCDSセンサを指で遮られると、光の量が少なくなり暗くなるのでロボットが左腕を上げるプログラミングです。）
	Distance	PSD Digital（距離センサ）：壁が一定距離近づくと、後退りして右折し、前進するプログラムです。 PSD Analog（距離センサ）：壁に近づくと左折しながら壁を回避するプログラムです。
	Dynamics	Accerlateration：倒れた時に立ち上がるプログラミングです。（前、後ろ）
Communication（通信）	IRRciever	リモコンの1番から8番に音の高さを設定して、DRCコントローラーのサウンドを鳴らすプログラムです。1～8番に”ド”～”シ”までマッチングします。（With Sound）
	Button	DRCコントローラーのボタンを押すと、コントローラーの裏側のLEDが反応するプログラムです。（with LED）

Moveの例

例の説明

MoveはDRCコントローラーに保存されたロボットのモーションを取得し、プログラムに適用します。ロボットのモーションも番号でインポートし、番号ごとの動きの名前はDR-SIMで確認することができます。 DR-SIMで編集された特定のモーションをロボットが無限に繰り返すプログラミングをします。ロボットの信頼性テストや展示などで活用度が高く、難易度が高いプログラムなので、一つずつ実行してください。

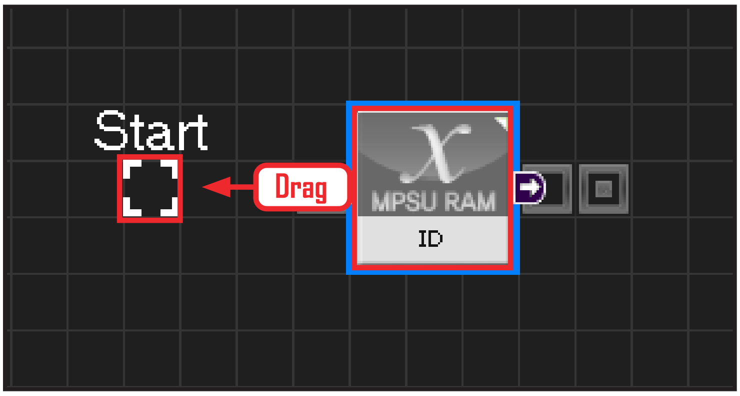
*この例で使用するモーションとモーション番号は基本提供するモーションとは異なります。 DR-SIMを使用して編集したモーションをDRCにダウンロードした場合を想定して作成された例です。関連モーションは www.hovis.co.kr/guide からダウンロードしてください。



01 変数の指定

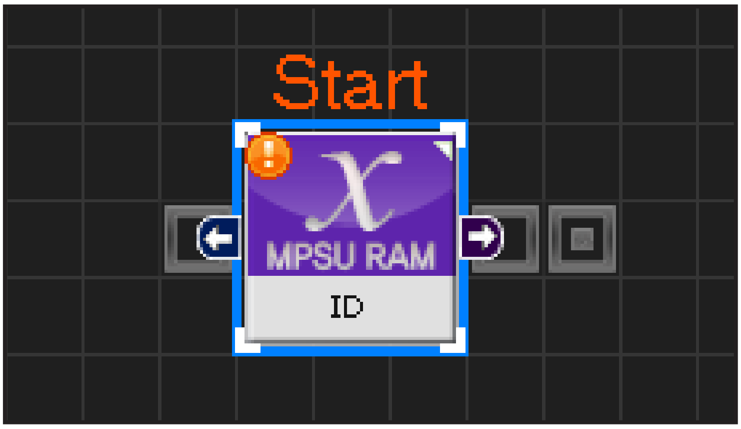
ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data > Variableモジュールをクリックします。



02 開始

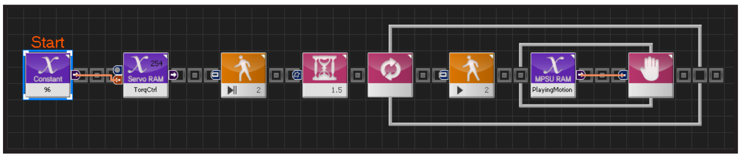
モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。



03 プログラミングの開始

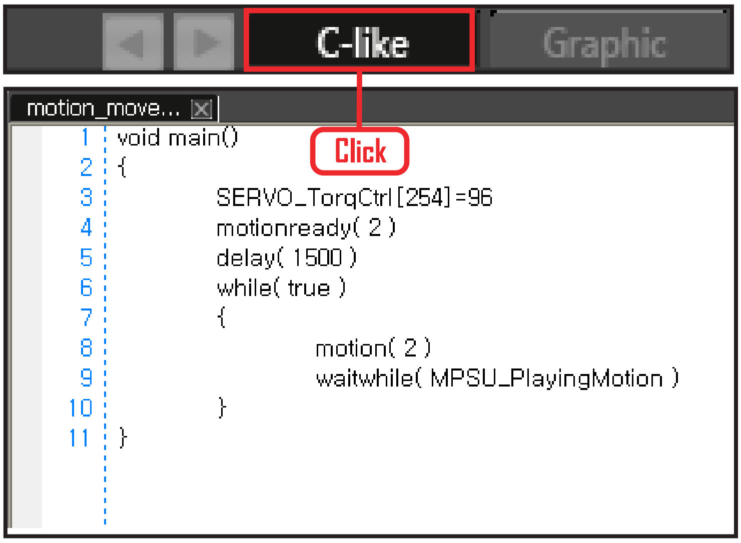
モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。

これはプログラミングが開始されたという意味です。


Start
Constant
96
254
Servo RAM
TorqCtrl
2
1.5
2
MPSU RAM
PlayingMotion

04 完全プログラミング

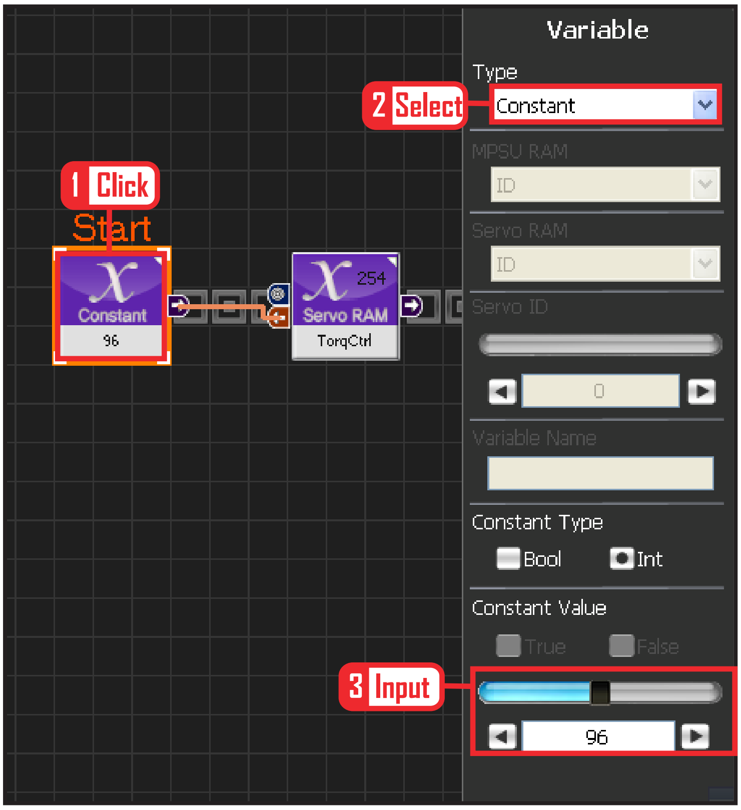
保存したモーションをインポートして繰り返させるプログラムです。Motion ready値に注意してください。


C-like
Graphic
motion_move...
Click

```
void main()
{
        SERVO_TorqCtrl[254]=96
        motionready( 2 )
        delay( 1500 )
        while( true )
        {
                motion( 2 )
                waitwhile( MPSU_PlayingMotion )
        }
}
```

05 C-Like

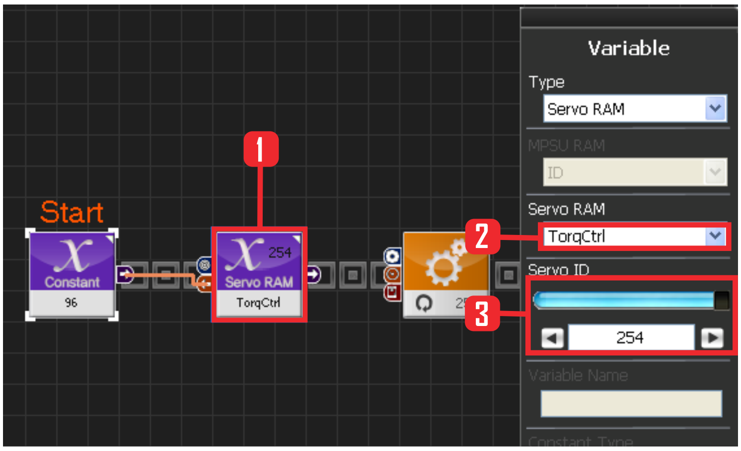
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
保存したモーションを利用して完全プログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。


1 Click
Start
Constant
96
254
Servo RAM
TorqCtrl
2 Select
3 Input
Variable
MPSU RAM
Servo RAM
Servo ID
Variable Name
Constant Value
☐True
☐False

Type: Constant
ID
ID
0
Constant Type: ☐Bool ●Int
96

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。

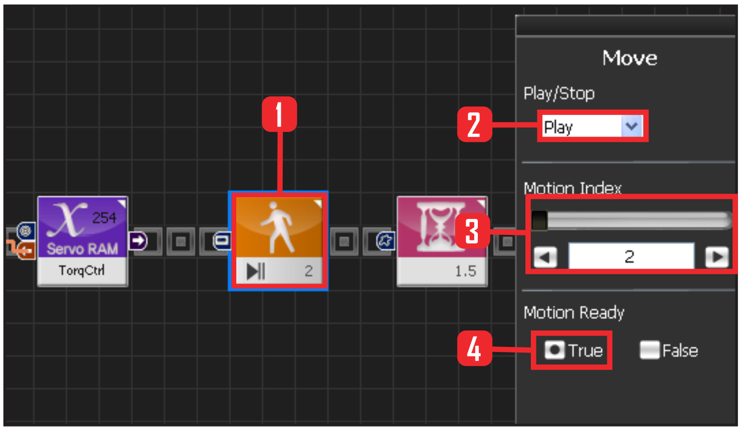


Variable
Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name
Constant Type

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type : Servo RAMを選択します。
Servo RAM : TorqCtrlを選択します。
Servo ID : 254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

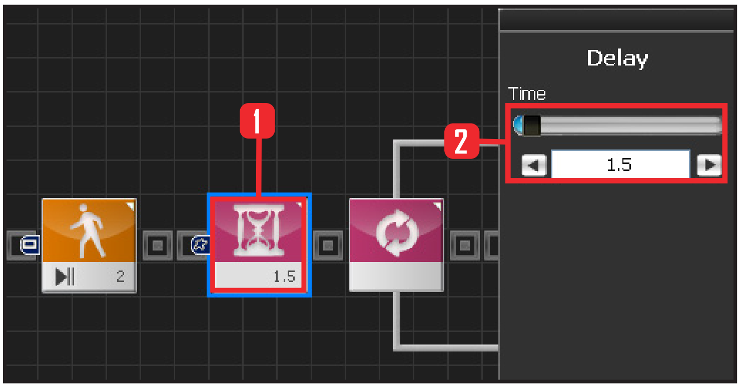


Move
Play/Stop: Play
Motion Index: 2
Motion Ready: ● True ☐ False

08 Motion Ready

保存したモーションをインポートする時、ロボットの現在状態で、急に動きが変動する場合があります。現在状態とモーションのスタート状態があまりにも違うとモーターに無理がかかり、ユーザーに危険が及びます。そこでMotion Ready、つまり、モーションの動作準備時間を設定する必要があります。

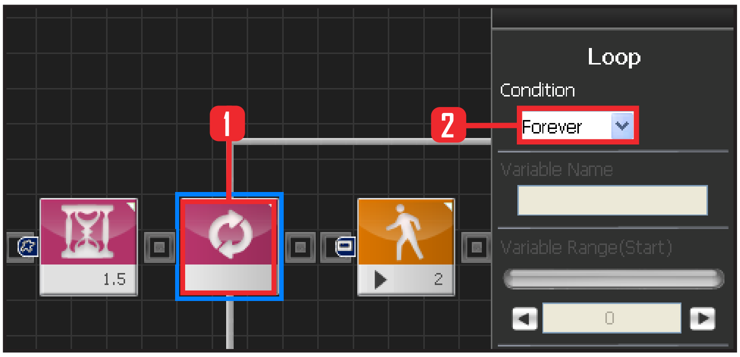
Motion > Moveを選択します。
Play/Stop : Playを選択します。
Motion Index : 2を選択します。 2番のモーションを持ってくるという意味です。ちなみにこのプログラムの2番モーションは座った後に立ち上がるモーションです。必ず2番のモーションを選択する必要はなくて、ユーザーが実行したいモーションの番号を入力してください。
Motion Ready : Trueを選択します。 Trueを選択すると、動作させようとするモーションの初期状態に徐々に移動します。



Delay
Time: 1.5

09 Delay

Motion Ready 動作が終了する前にモーションが進行することを防止するためにDelay値を1.5秒に設定します。



Loop

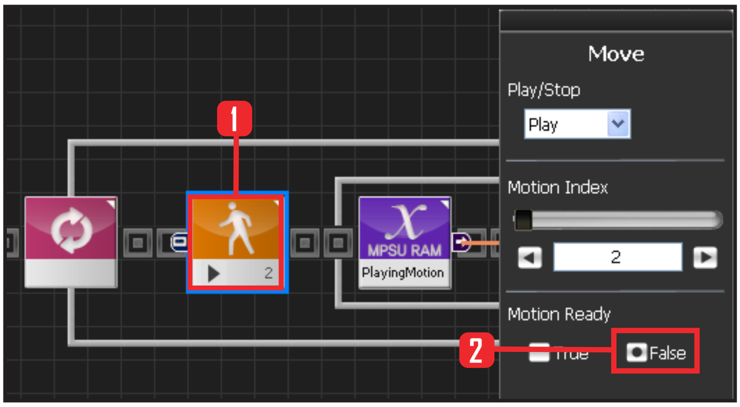
Condition: Forever

Variable Name

Variable Range(Start): 0

10 繰り返し

Forever無限ループを設定します。



Move

Play/Stop: Play

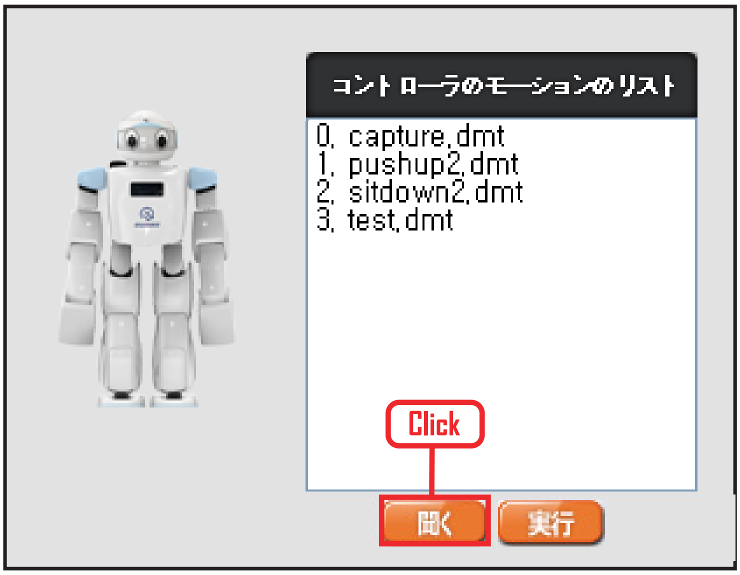
Motion Index: 2

Motion Ready: ○True ●False

11 モーション動作

Motion　Ready値をFalseに設定すると、モーションを最初から最後まで動作させるという意味になります。

Motion Ready : Falseを選択します。



コントローラのモーションのリスト

0. capture.dmt
1. pushup2.dmt
2. sitdown2.dmt
3. test.dmt

参照 : モーションを見る

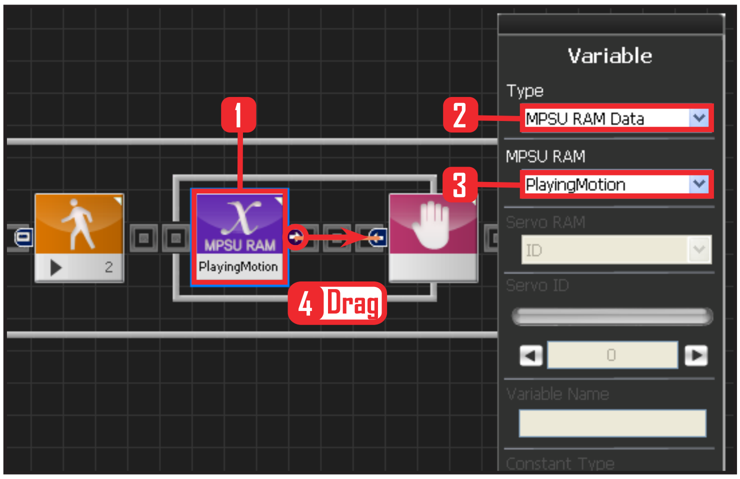
DR-SIMプログラムでロボットと接続して、ロボット設定をクリックすると、現在のコントローラーにあるロボットのモーションを確認できます。

現在、2番のモーションは両腕を広げて座った後、立ち上がるモーションです。

12 モーション動作の確認

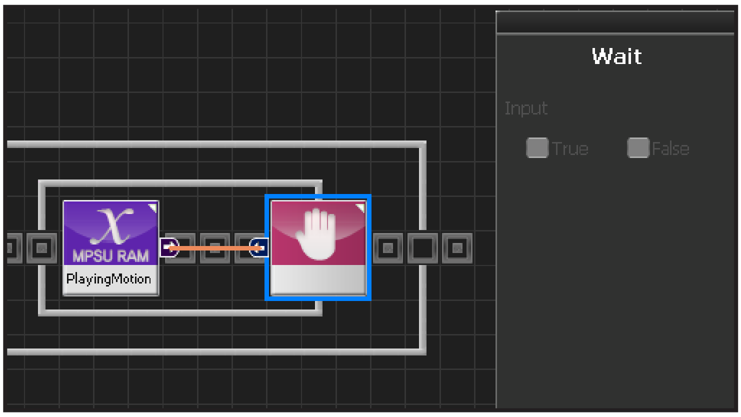
Loopは継続的な繰り返しを意味します。 Move命令を下してから、実際のモーションが実行され、完了するまでには時間がかかるので、loop内にMoveモジュールを一つだけ入れて実行すると、モーションが既に実行中であるにもかかわらずloopを続けてモーションを実行する命令を繰り返すことになります。
この場合、Moveモジュールを実行した回数と実際のモーションを実行する回数が異なります。
したがって実行したモーションが終わるまで待ってから再びloopの最初に戻す方がより正確です。
Variable > MPSU RAM Dataに入れれば、Playing Motion があります。 Playing Motionはロボットのモーションを実行しているかを確認する変数です。そのPlaying MotionにWaitをかければ、ロボットの動作が終わるまでLoopを待ちます。

Data > Variableモジュールを選択します。
Type：MPSU RAM Dataを選択します。
MPSU RAM：Playing Motionを選択します。
Outputコネクタの値の後ろのWaitモジュールに接続します。
モーションが終わるまで待つことを意味します。


Variable
Type: MPSU RAM Data
MPSU RAM: PlayingMotion
Servo RAM: ID
Servo ID: 0
Variable Name
Constant Type
1
2
3
4 Drag
MPSU RAM
PlayingMotion
2

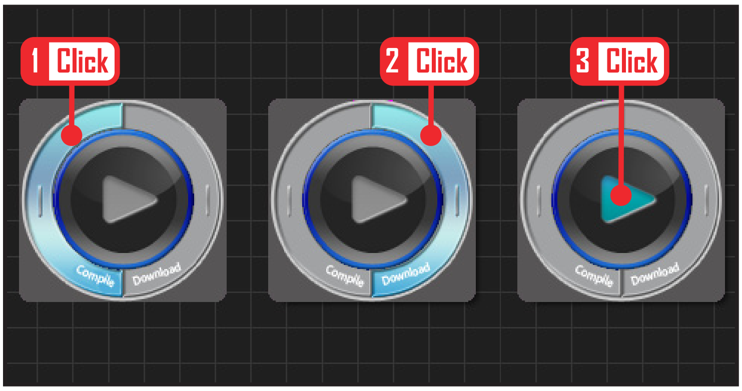
13 Wait

モーションが終わるまで待ちます。
モーションが終わればまた最初に戻ってモーションを繰り返します。

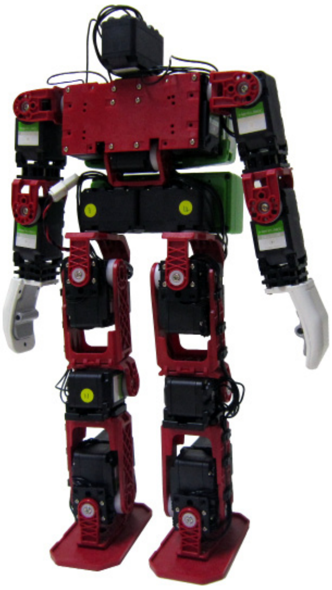

Wait
Input
☐True ☐False
MPSU RAM
PlayingMotion

14 コンパイル、ダウンロード、実行

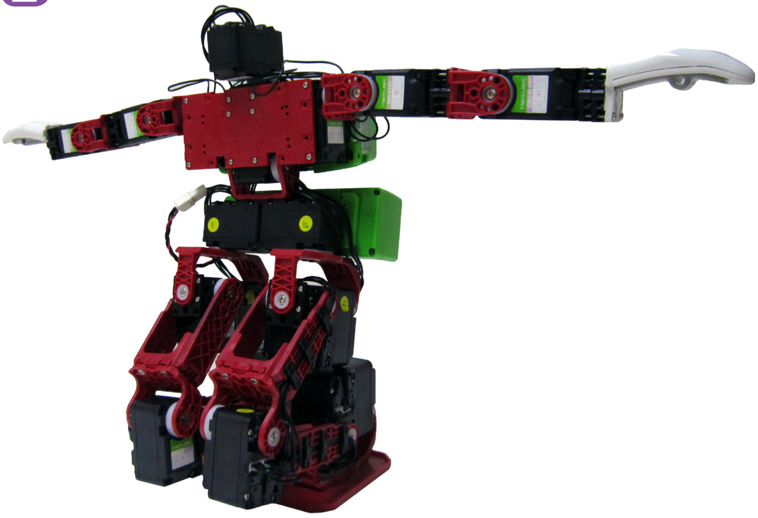
左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

1



2



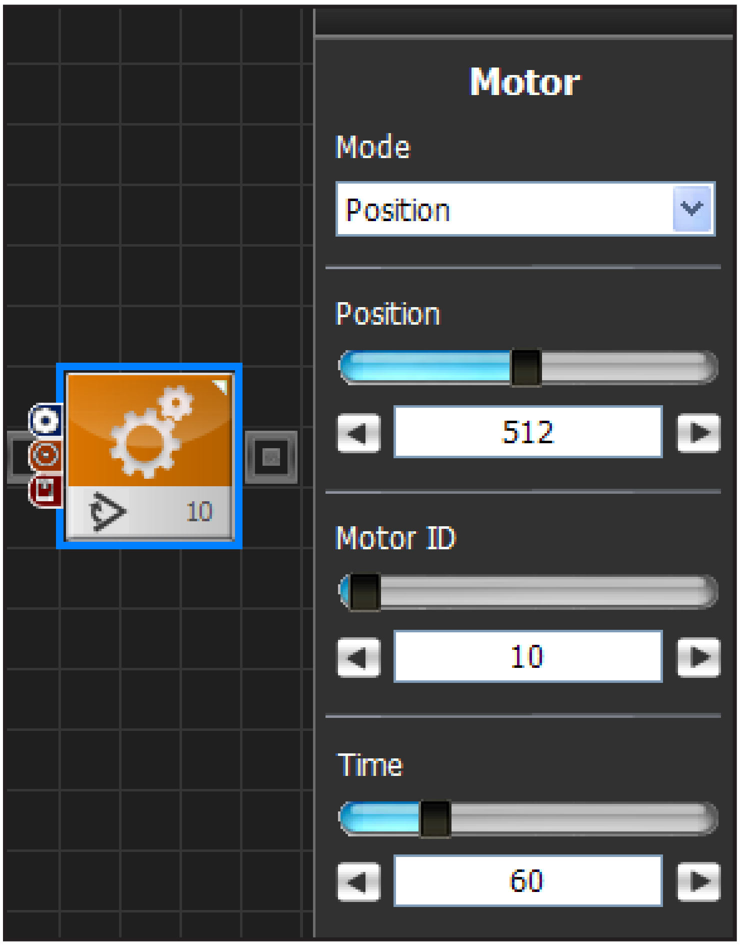
15 ロボット動作

ロボットが座って立ち上がるモーションを繰り返し実行します。

モジュール別のプログラミング：モーション(Motion) > Motor

Motor 説明

Motorモジュールは2つの動作モードがあります。一つは位置制御モード、もう1つは速度制御モードです。



Motor

Mode: Position

Position: 512

Motor ID: 10

Time: 60

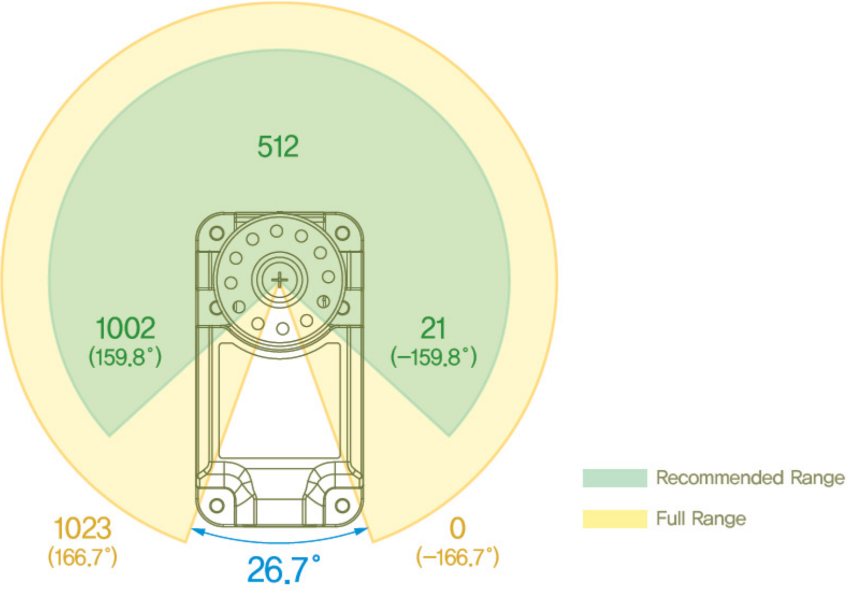
❶ 位置制御モード

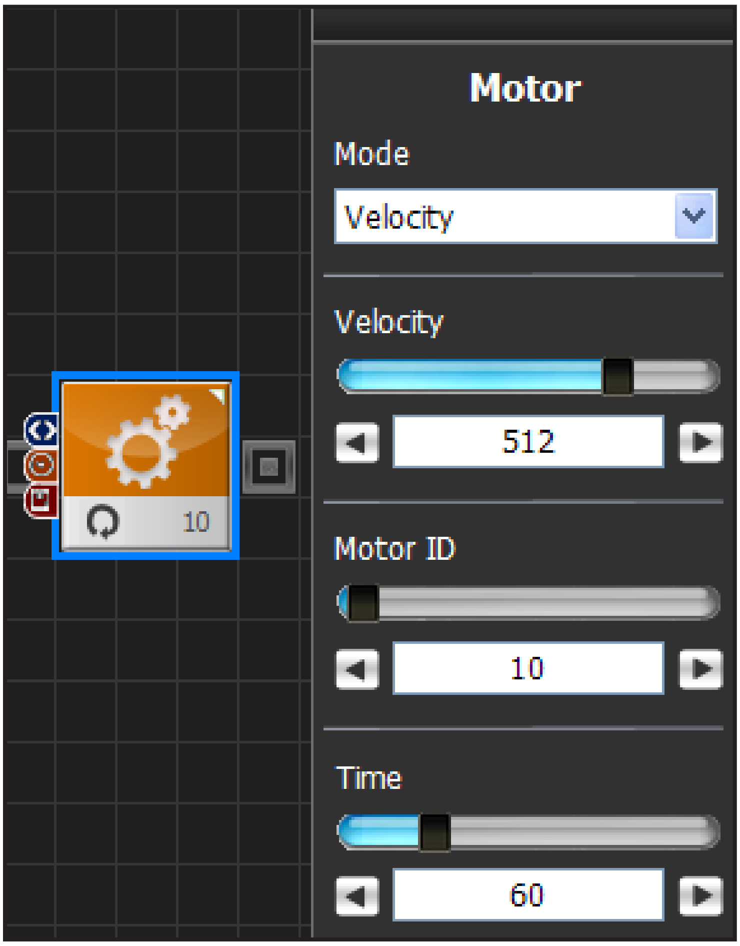
位置制御モードはモーターを特定の位置に移動させるモードです。

Positionの範囲は-127～1152の値です。サーボの工場出荷時の初期設定で許可される値は21～1002であり、この範囲を超える値はモーターの最小/最大の位置と位置補正値が変更されると自動的に変更されます。モーターの定位置の値は512で、組み立てる際の基準となる位置も512です。通常Hovisのすべてのモーターの位置を512に設定すると、両腕を90度に広げて立った姿勢になります。サーボの制御範囲と定位置に関して下図を参照してください。

Motor IDは制御するサーボのIDです。

Timeはサーボの目標位置に移動する時間です。 1あたり11.2msの実際時間を意味します。例えば100に設定すると、1.12秒間かけて設定した位置に移動します。


512
1002
(159.8˚)
21
(–159.8˚)
1023
(166.7˚)
26.7˚
0
(–166.7˚)
Recommended Range
Full Range



Motor

Mode: Velocity

Velocity: 512

Motor ID: 10

Time: 60

2 速度制御モード

速度制御モードはモーターを指定速度で無限に回転させるモードです。

Velocityの範囲は-1023～1023の値です。数字が大きいほど高速で回転させることができます。正と負の違いは回転方向です。

Motor IDは制御するサーボのIDです。

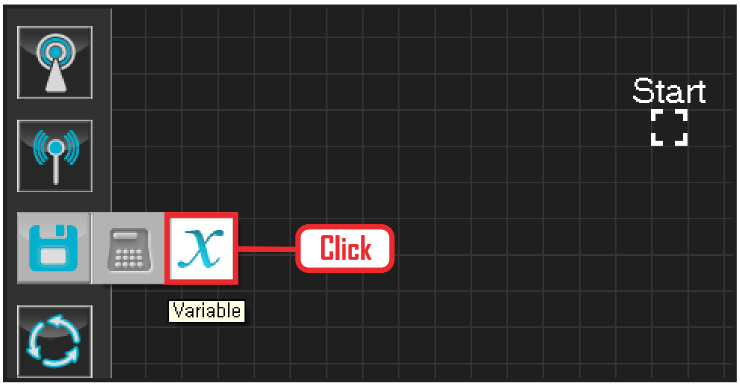
Timeはサーボが目標速度に到達する時間です。 1あたり11.2msの実際の時間を意味します。例えば100に設定すると、1.12秒間かけて目標の速度に向けて徐々に速度が変化します。

Motorの例

例の説明

一般的にロボットのモーションはモーターを一つずつ制御し、総合して動作を作成します。DR-SIM（Motion　Editor）を使うと、複雑なモーション作成をを簡単に行うことができます。

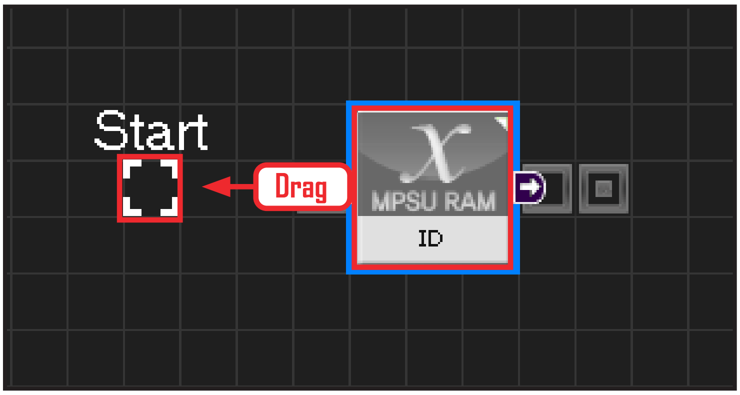
今回のプログラムではMotion　Editorを使用せずにTask　Editorを利用して、モーターを一つずつ操作して、連続するモーションを作ります。モーションの目標はウェーブダンスを踊るロボットです。完了すると、おもしろい動作をするので、是非挑戦してみてください。



01 変数の指定

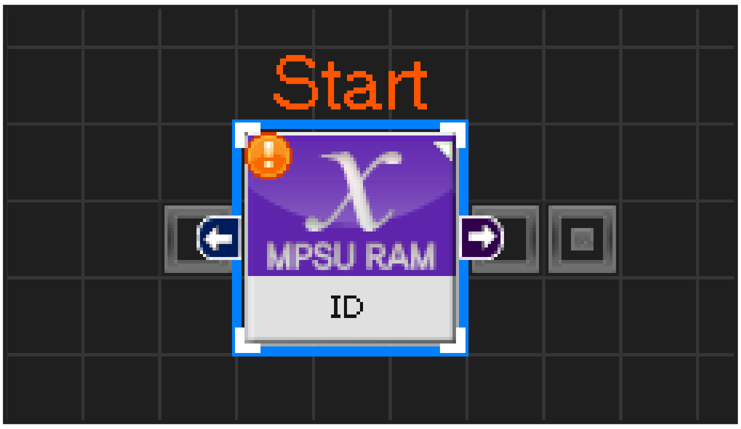
ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data > Variableモジュールをクリックします。



02 開始

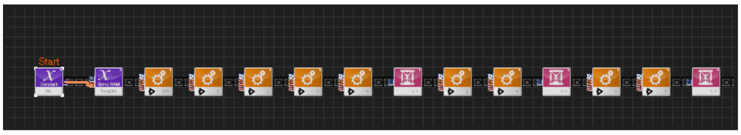
モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。

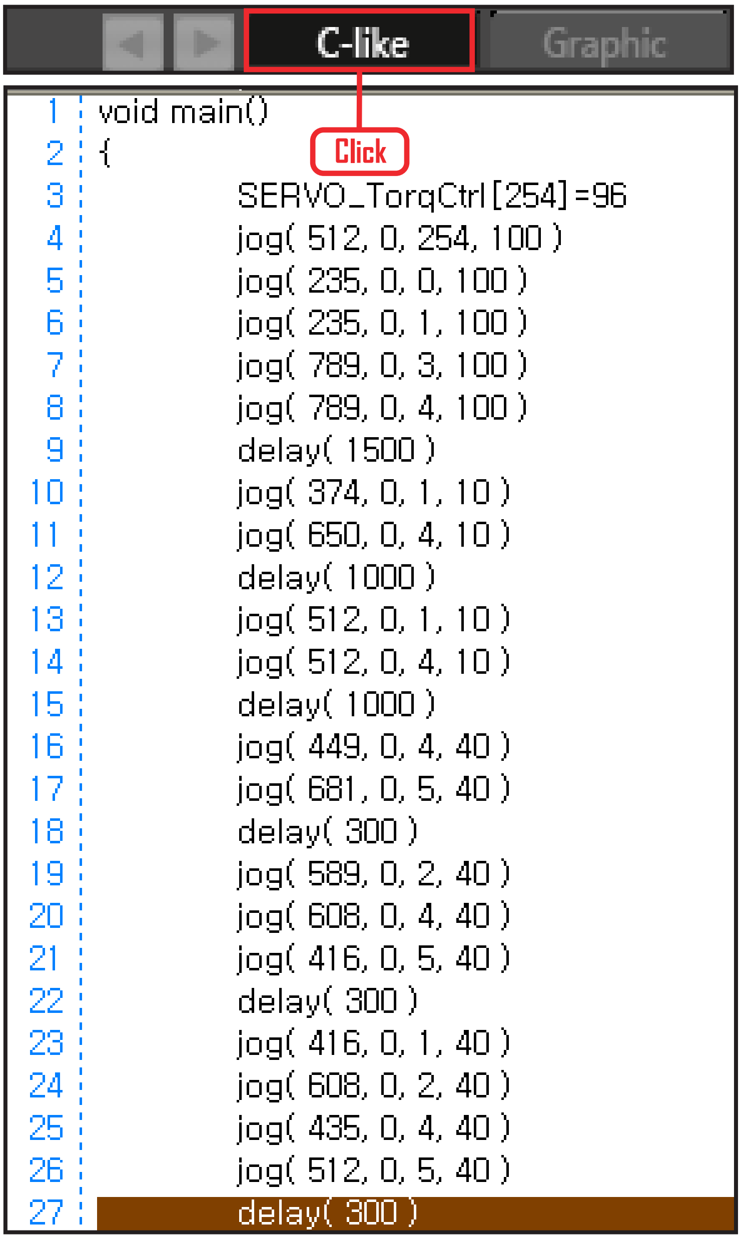


03 プログラミングの開始

モジュールとStart　Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。

これはプログラミングが開始されたという意味です。





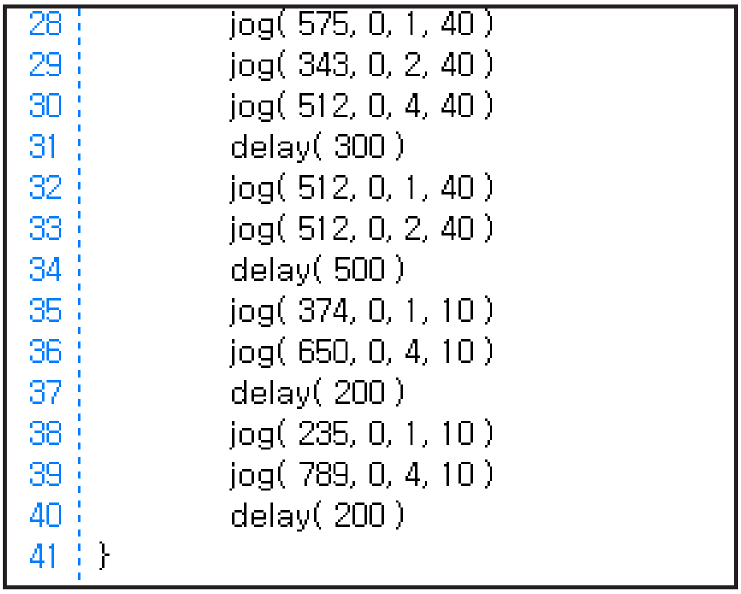
```
void main()
{
        SERVO_TorqCtrl[254]=96
        jog( 512, 0, 254, 100 )
        jog( 235, 0, 0, 100 )
        jog( 235, 0, 1, 100 )
        jog( 789, 0, 3, 100 )
        jog( 789, 0, 4, 100 )
        delay( 1500 )
        jog( 374, 0, 1, 10 )
        jog( 650, 0, 4, 10 )
        delay( 1000 )
        jog( 512, 0, 1, 10 )
        jog( 512, 0, 4, 10 )
        delay( 1000 )
        jog( 449, 0, 4, 40 )
        jog( 681, 0, 5, 40 )
        delay( 300 )
        jog( 589, 0, 2, 40 )
        jog( 608, 0, 4, 40 )
        jog( 416, 0, 5, 40 )
        delay( 300 )
        jog( 416, 0, 1, 40 )
        jog( 608, 0, 2, 40 )
        jog( 435, 0, 4, 40 )
        jog( 512, 0, 5, 40 )
        delay( 300 )
```

04 完全プログラミング

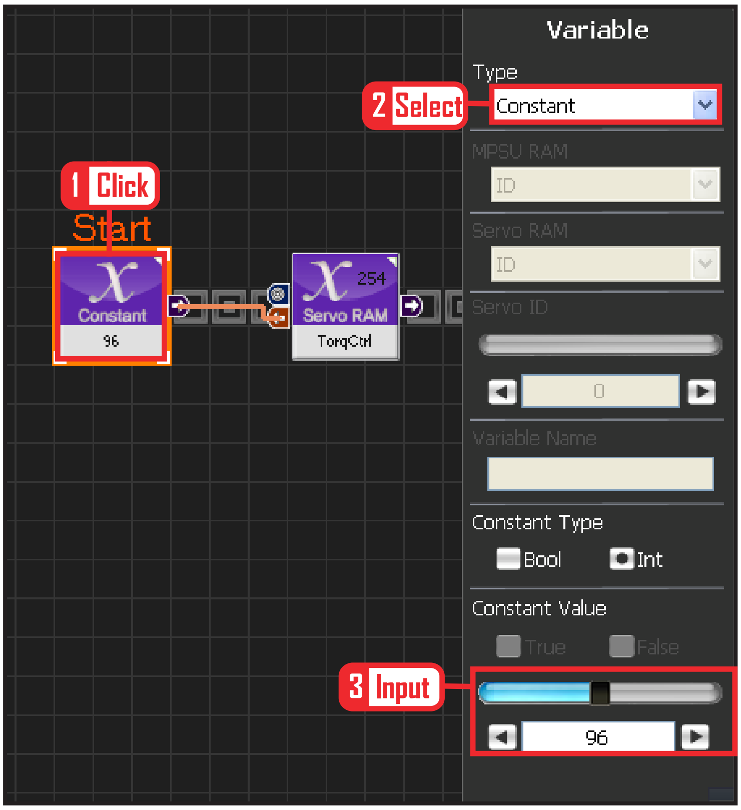
モーターを制御する全体のプログラミングです。

05 C-Like

右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
モーターモジュールを利用して完全プログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。



```
            jog( 575, 0, 1, 40 )
            jog( 343, 0, 2, 40 )
            jog( 512, 0, 4, 40 )
            delay( 300 )
            jog( 512, 0, 1, 40 )
            jog( 512, 0, 2, 40 )
            delay( 500 )
            jog( 374, 0, 1, 10 )
            jog( 650, 0, 4, 10 )
            delay( 200 )
            jog( 235, 0, 1, 10 )
            jog( 789, 0, 4, 10 )
            delay( 200 )
}
```



1 Click

Start

2 Select

Type: Constant

3 Input

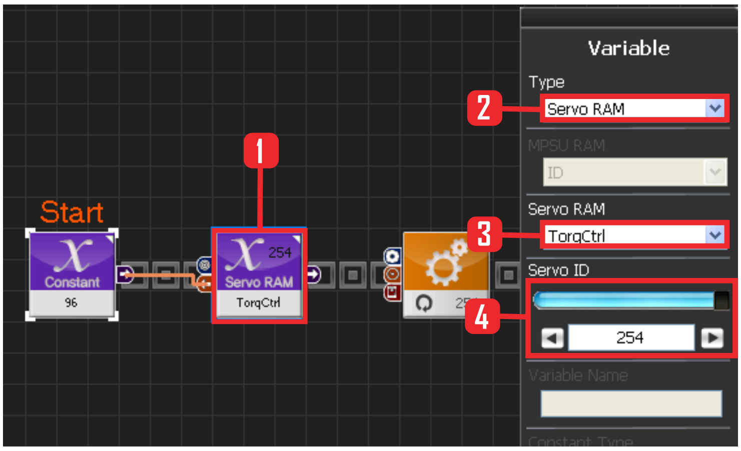
Constant Type: ☐Bool ●Int

Constant Value: 96

Constant
96
254
Servo RAM
TorqCtrl
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
True
False

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant Value値を96に設定します。サーボのTorqControlレジスタに96(0x60)という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。

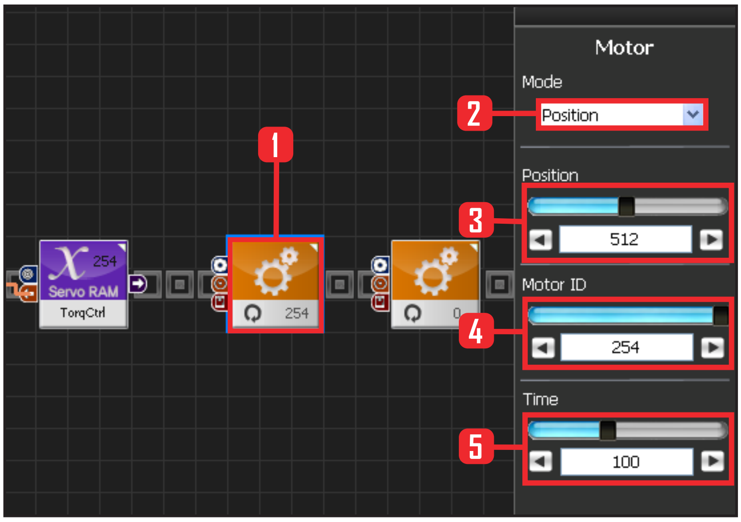


Variable
Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name
Constant Type

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type：Servo RAMを選択します。
Servo RAM：TorqCtrlを選択します。
Servo ID：254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

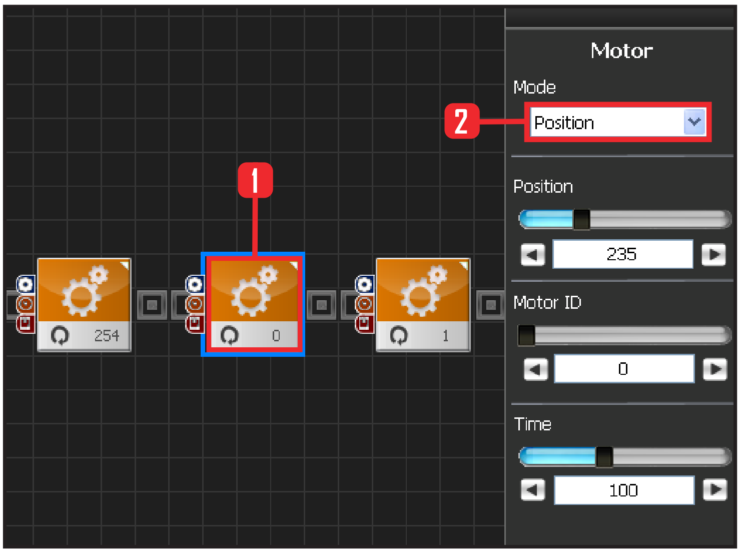


Motor
Mode: Position
Position: 512
Motor ID: 254
Time: 100

08 すべてのサーボモーターの角度を設定

すべてのサーボモーターの角度を中央に移動する方法です。

Motion > Motorを選択します。
Mode：Positonを選択します。角度を調整します。
Position：512に設定します。512番はモーターを中央に移動するという意味です。
Motor ID：254に設定します。 254はすべてのモーターに適用するという意味です。
Time：100に設定します。単位は1あたり11.2msで、100は約1.12秒を意味します。
1.12秒間かけて設定角度に移動させるという意味です。

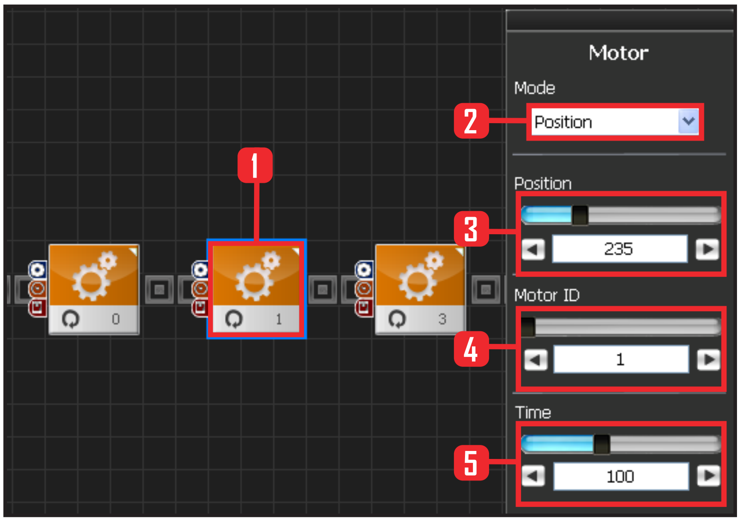


Motor
Mode: Position
Position: 235
Motor ID: 0
Time: 100

09 モーター0番（右肩）の設定

1段階：初期姿勢
直立不動の姿勢を作成します。
すべてのロボットのモーターの角度を中央に整列させると、ヒューマノイドは腕を左右に広げます。これを直立不動の姿勢にさせるモーションを容易に作成することができます。

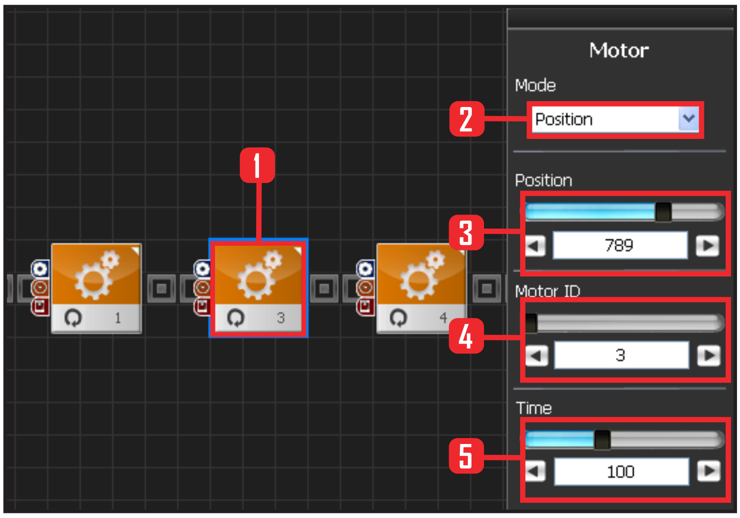
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：235に設定します。 235は水平に広げた右手を垂直に移動させるようにモーターを回すことです。
Motor ID：0に設定します。右肩のモーターIDが0番です。
Time：100に設定します。約1.12秒間設定角度に移動します。



Motor
Mode: Position
Position: 235
Motor ID: 1
Time: 100

10 モーター1番（右腕の上）の設定

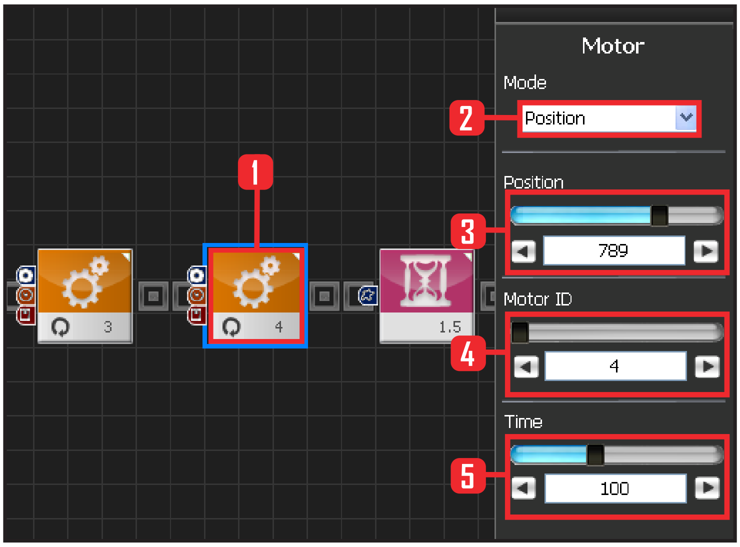
Mode：Postionを選択します。
Position：235を選択します。 235は水平に広げた腕を垂直に下げる設定です。
Motor ID：1に設定します。右上の腕の肩に接続しているモーターが1番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。



Motor
Mode: Position
Position: 789
Motor ID: 3
Time: 100

11 モーター3番（左肩）の設定

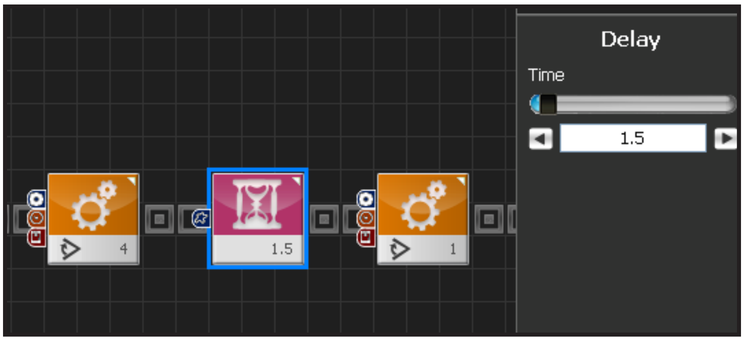
Mode：Postionを選択します。
Position：789を選択します。 789は水平に広げた左手を垂直に下げる位置です。
Motor ID：3に設定します。左側の肩のモーターが3番です。
Time：100に設定します。約1.12秒間設定角度に移動します。



Motor
Mode: Position
Position: 789
Motor ID: 4
Time: 100

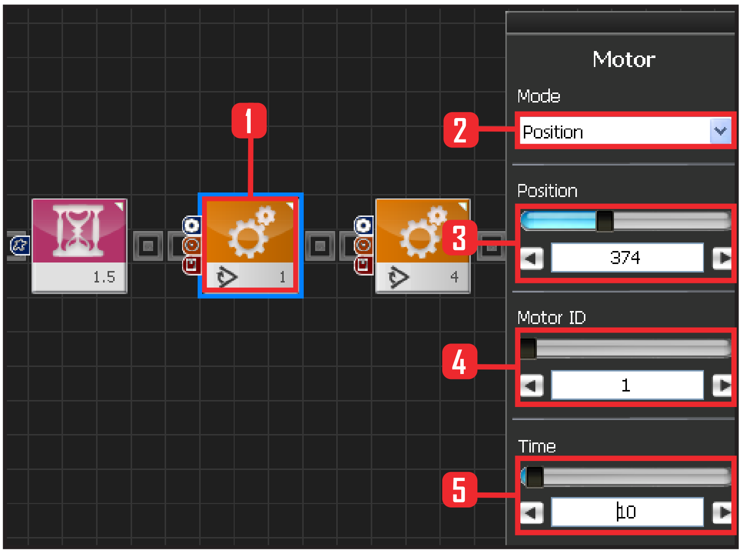
12 モーター4番（左腕の上）の設定

Mode：Postionを選択します。
Position：789を選択します。 789は水平に広げた手を垂直に下げる設定です。
Motor ID：4に設定します。左上の腕の肩に接続しているモーターが4番です。
Time：100に設定します。約1.12秒間設定角度に移動します。


Delay
Time
1.5

13 Delay 遅延させる

1.5秒間待ちます。

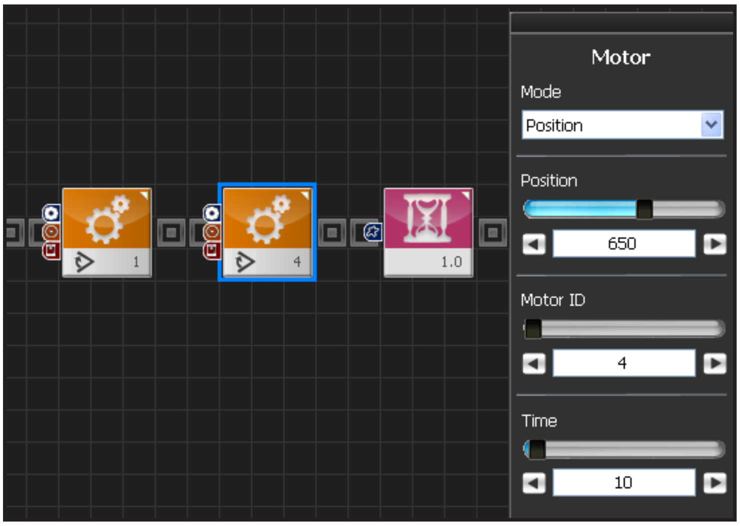


Motor
Mode
Position
Position
374
Motor ID
1
Time
10

14 モーター1番（右腕の上）の設定

2段階：45度の角度に腕を広げます。
ロボットが踊るための準備動作として腕を45度に設定します。

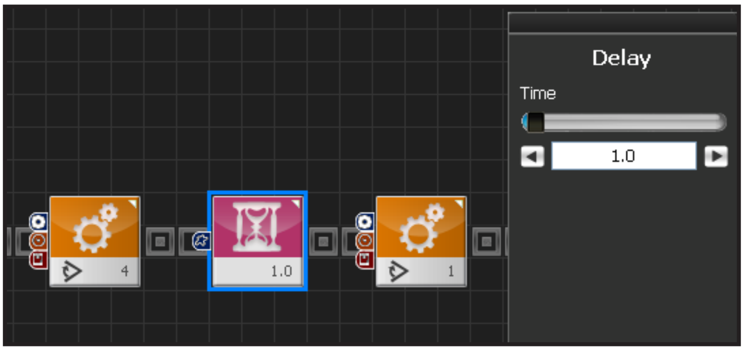
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：374に設定します。 374は右腕を45度に移動させます。
Motor ID：1に設定します。右腕の上のモーターIDが1番です。
Time：10に設定します。約0.112秒間かけてかけて設定角度に移動します。


Motor
Mode
Position
Position
650
Motor ID
4
Time
10

15 モーター4番（左腕の上）の設定

左腕の上のモーターを45度に設定します。

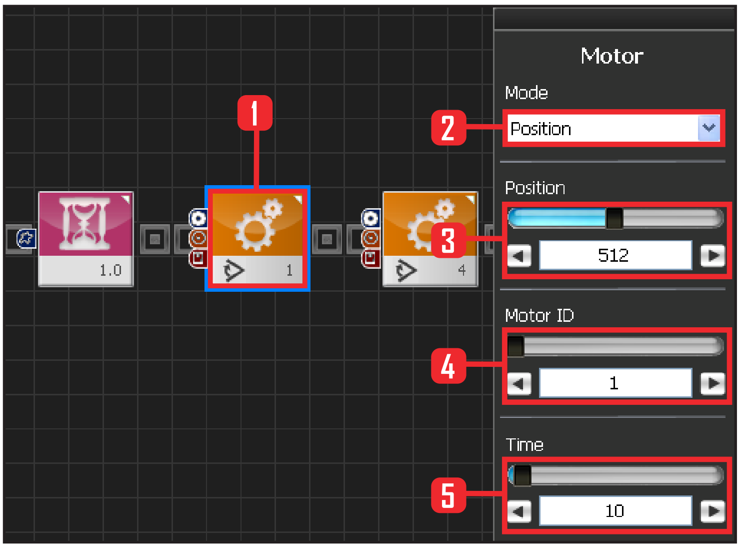
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：650に設定します。 650は右腕を45度に移動させます。
Motor ID：4に設定します。左腕の上のモーターIDが4番です。
Time：10に設定します。約0.112秒間かけて設定角度に移動します。



Delay
Time
1.0

16 Delay　遅延させる

1.0秒間待ちます。

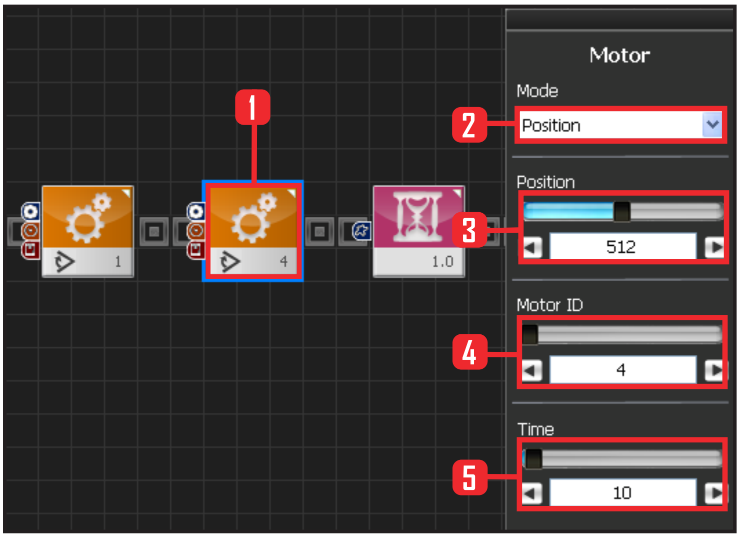


Motor
Mode
Position
Position
512
Motor ID
1
Time
10

17 モーター1番（右腕の上）の設定

3段階：90度の角度に腕を広げます。
ロボットの腕を90度に上げてウェーブダンスの段階に入ります。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：512に設定します。　512は腕を90度に移動させます。（モーターの中央の値）
Motor　ID：1に設定します。右腕の上のモーターIDが1番です。
Time：10に設定します。約0.112秒間かけてかけて設定角度に移動します。

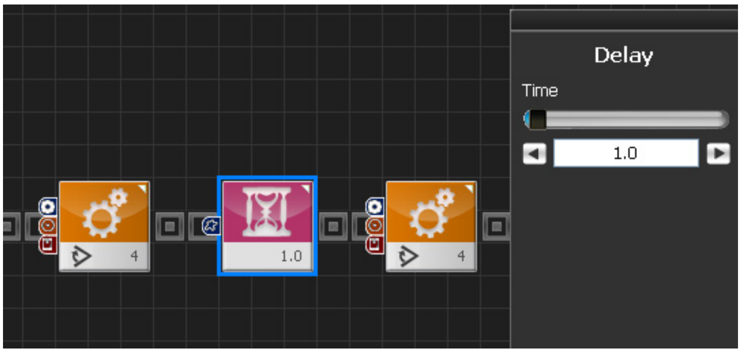


Motor
Mode
Position
Position
512
Motor ID
4
Time
10

18 モーター4番（左腕の上）の設定

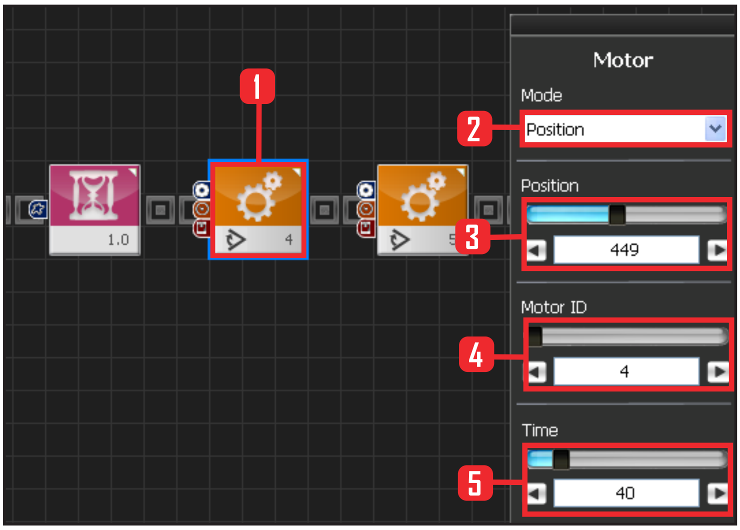
ロボットの腕を90度に上げてウェーブダンスの段階に入る

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：512に設定します。　512は腕を90度に移動させます。（モーターの中央の値）
Motor　ID：4に設定します。左腕の上のモーターIDが4番です。
Time：10に設定します。約0.112秒間かけてかけて設定角度に移動します。


Delay
Time
1.0

19 Delay 遅延させる

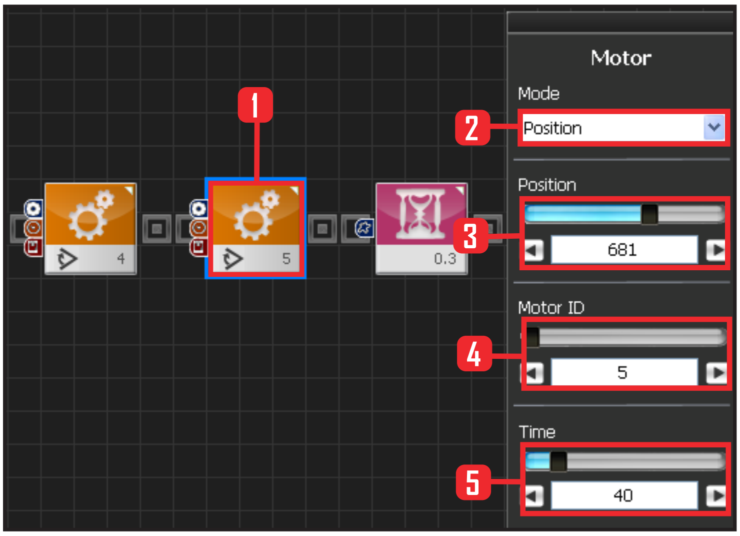
1.0秒間待ちます。


Motor
Mode
Position
Position
449
Motor ID
4
Time
40

20 モーター4番(左腕の上)の設定

4段階:ウェーブ1段階
左腕からウェーブを開始します。

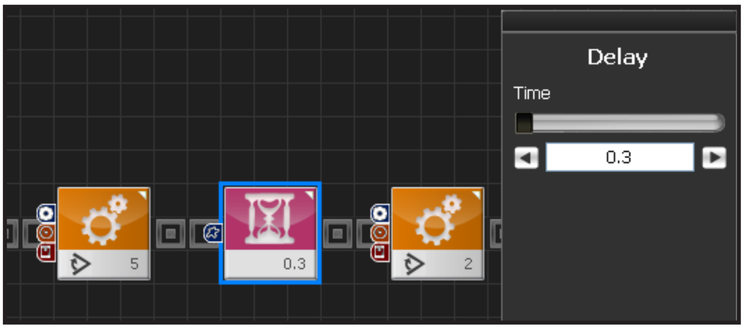
Motion > Motorを選択します。
Mode:Positionを選択します。
Position:449に設定します。449は腕をウェーブの開始位置に移動させます。
Motor ID:4に設定します。左腕の上のモーターIDが4番です。
Time:40に設定します。約0.448秒間かけて設定角度に移動します。


Motor
Mode
Position
Position
681
Motor ID
5
Time
40

21 モーター5番(左腕の下)の設定

左腕の下のウェーブを開始します。

Motion > Motorを選択します。
Mode:Positionを選択します。
Position:681に設定します。
Motor ID:5に設定します。左腕の下のモーターIDが5番です。
Time:40に設定します。約0.448秒間かけて設定角度に移動します。



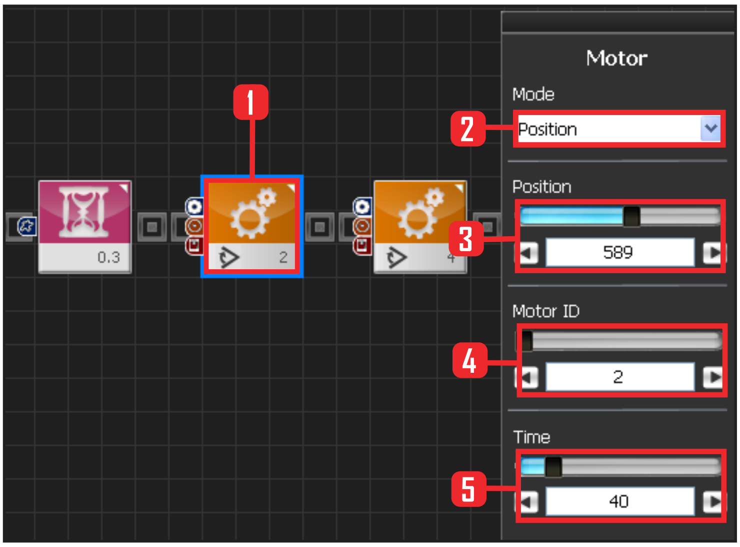
Delay

Time

0.3

22 Delay 遅延させる

0.3秒間待ちます。
踊りのモーション部分なので、Delay Timeを短く設定します。



Motor

Mode

Position

Position

589

Motor ID

2

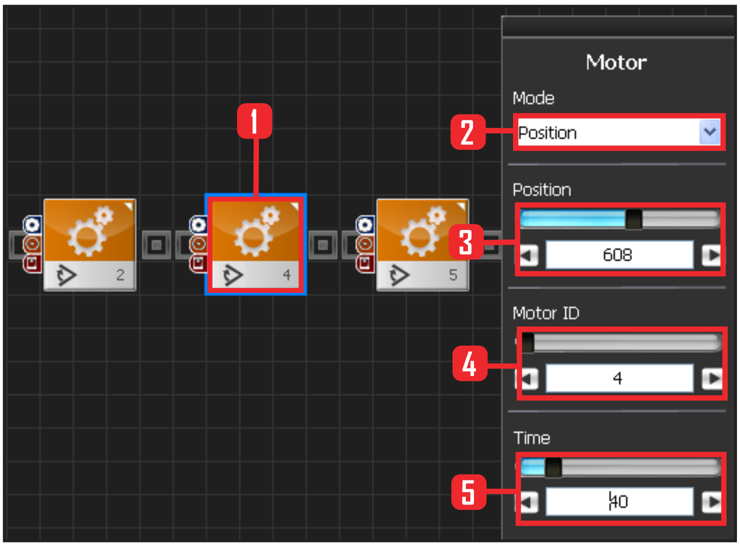
Time

40

23 モーター2番（右腕の下）の設定

5段階：ウェーブ2段階
左腕の下のウェーブを開始します。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：589に設定します。
Motor ID：2に設定します。右腕の下のモーターIDが2番です。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。



Motor

Mode

Position

Position

608

Motor ID

4

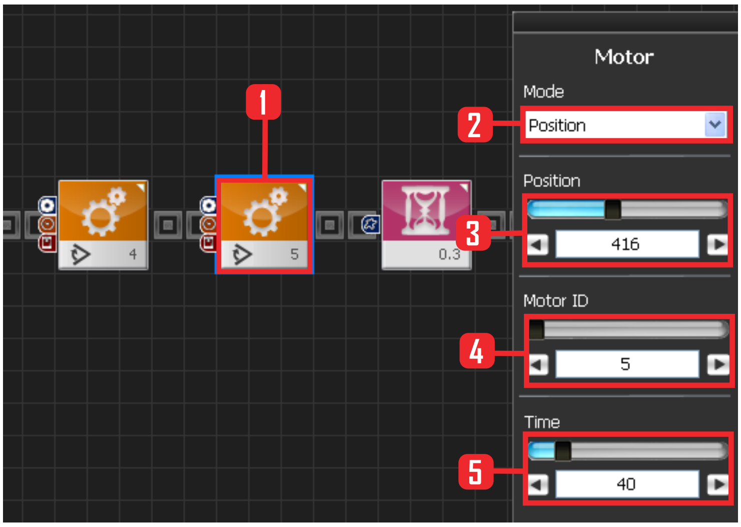
Time

40

24 モーター4番（左腕の上）の設定

モーターの角度を少しずつ変化させ、ウェーブ効果を与えます。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：608に設定します。
Motor ID：4に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。

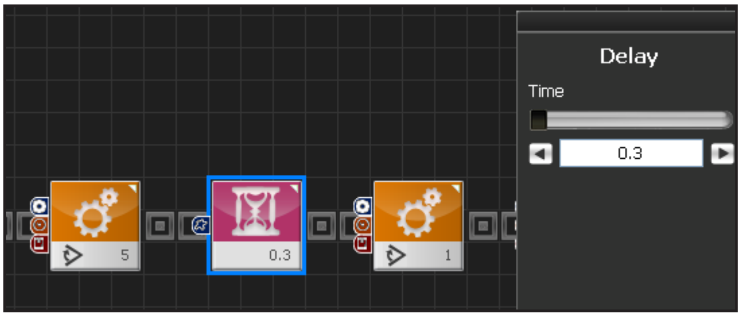


Motor
Mode
Position
Position
416
Motor ID
5
Time
40

25 モーター5番（左腕の下）の設定

モーターの角度を少しずつ変化させ、ウェーブ効果を与えます。

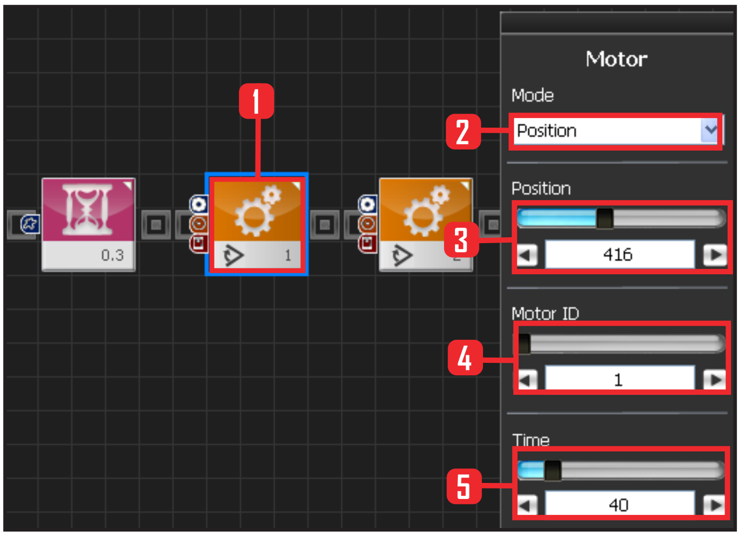
Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 416に設定します。
Motor ID : 5に設定します。
Time : 40に設定します。約0.448秒間かけて設定角度に移動します。



Delay
Time
0.3

26 Delay 遅延させる

0.3秒間待ちます。
踊りのモーション部分なので、Delay Timeを短く設定します。

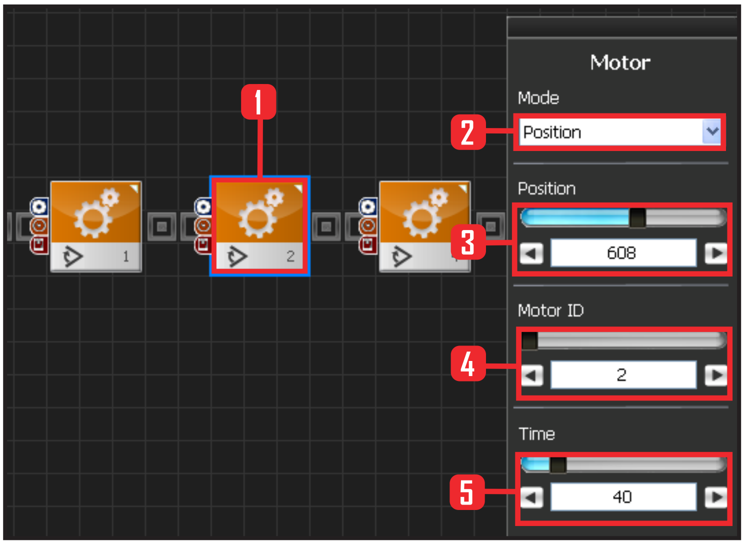


Motor
Mode
Position
Position
416
Motor ID
1
Time
40

27 モーター1番（右腕の上）の設定

6段階 : ウェーブ3段階
ウェーブの状態から元に戻します。

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 416に設定します。
Motor ID : 1に設定します。
Time : 40に設定します。約0.448秒間かけて設定角度に移動します。

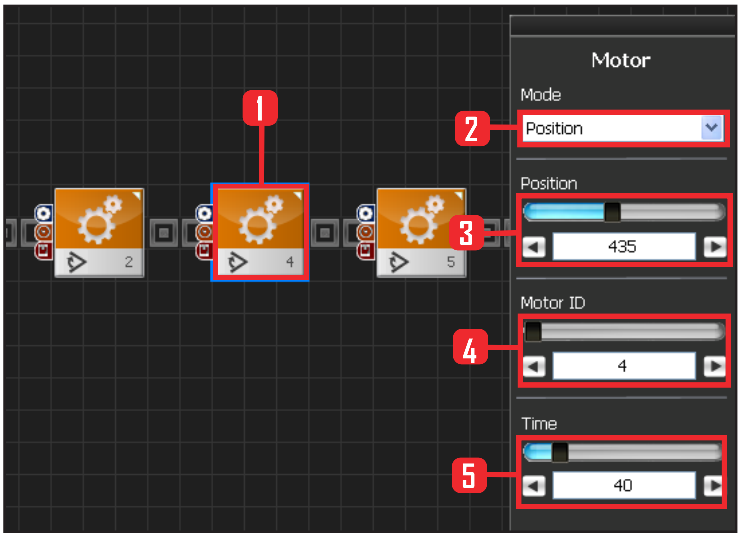


Motor
Mode
Position
Position
608
Motor ID
2
Time
40

28 モーター2番（右腕の下）の設定

ウェーブの状態から元に戻します。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：608に設定します。
Motor ID：2に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。

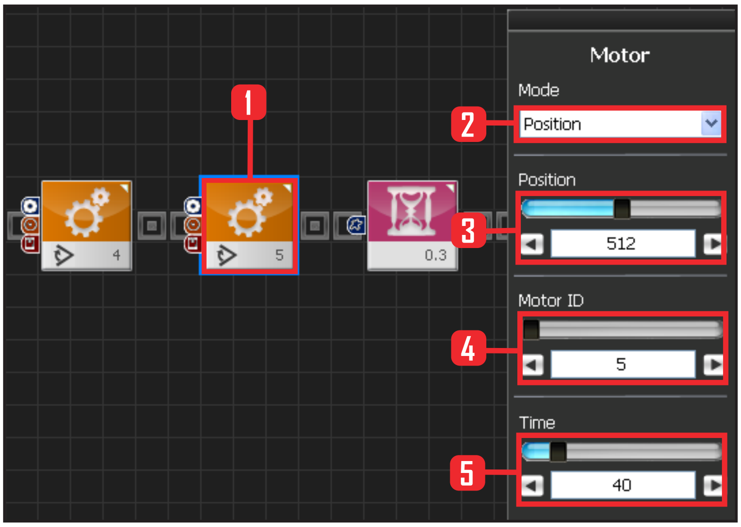


Motor
Mode
Position
Position
435
Motor ID
4
Time
40

29 モーター4番（左腕の上）の設定

ウェーブの状態から元に戻します。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：435に設定します。
Motor ID：4に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。

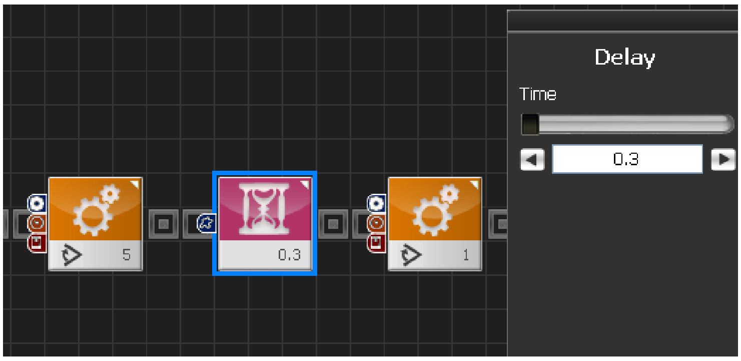


Motor
Mode
Position
Position
512
Motor ID
5
Time
40

30 モーター5番（左腕の下）の設定

ウェーブの状態から元に戻します。

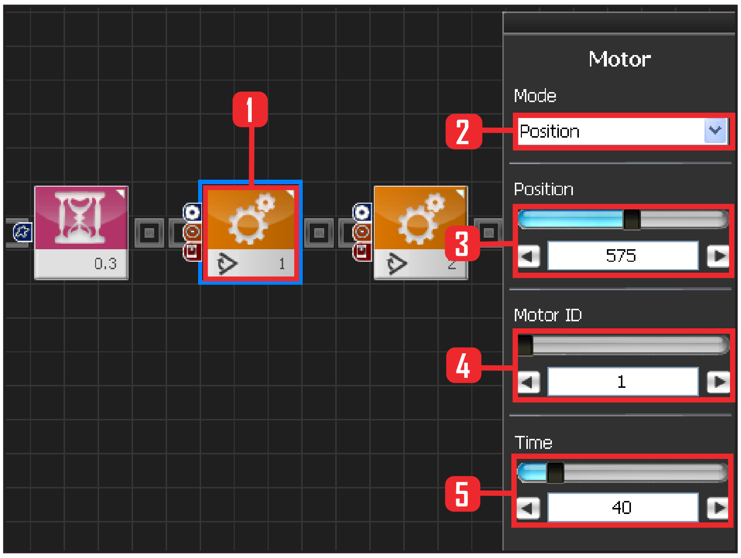
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：512に設定します。
Motor ID：5に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。



Delay
Time
0.3

31 Delay 遅延させる

0.3秒間待ちます。
踊りのモーション部分なので、Delay Timeを短く設定します。

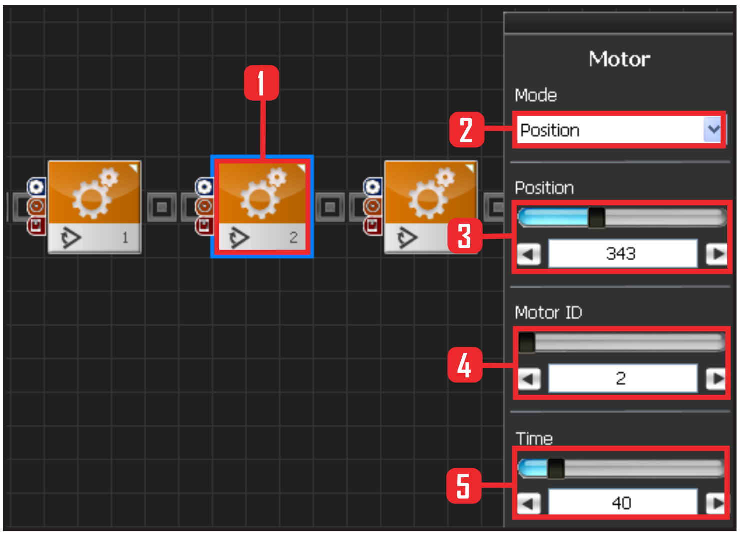


Motor
Mode
Position
Position
575
Motor ID
1
Time
40

32 モーター1番（右腕の上）の設定

7段階：ウェーブ4段階
ウェーブを終わります。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：575に設定します。
Motor ID：1に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。

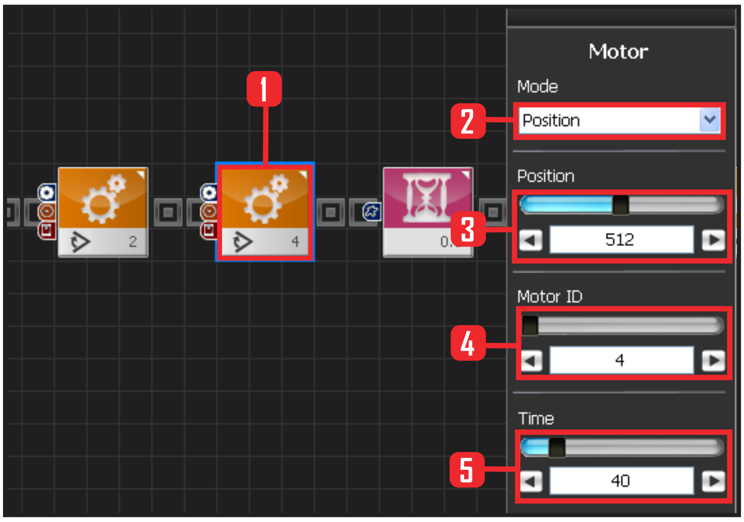


Motor
Mode
Position
Position
343
Motor ID
2
Time
40

33 モーター2番（右腕の下）の設定

ウェーブを終わります。

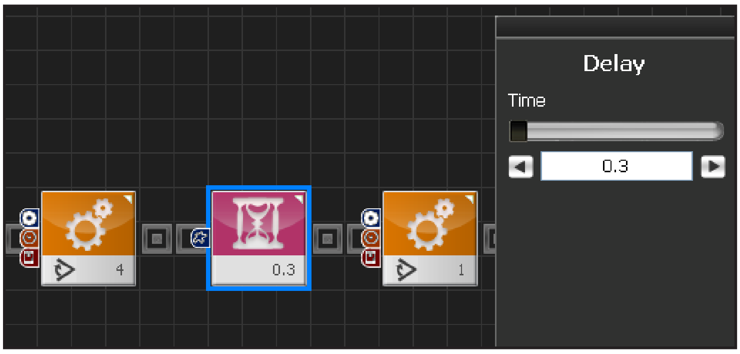
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：343に設定します。
Motor ID：2に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。


Motor
Mode: Position
Position: 512
Motor ID: 4
Time: 40

34 モーター4番（左腕の上）の設定

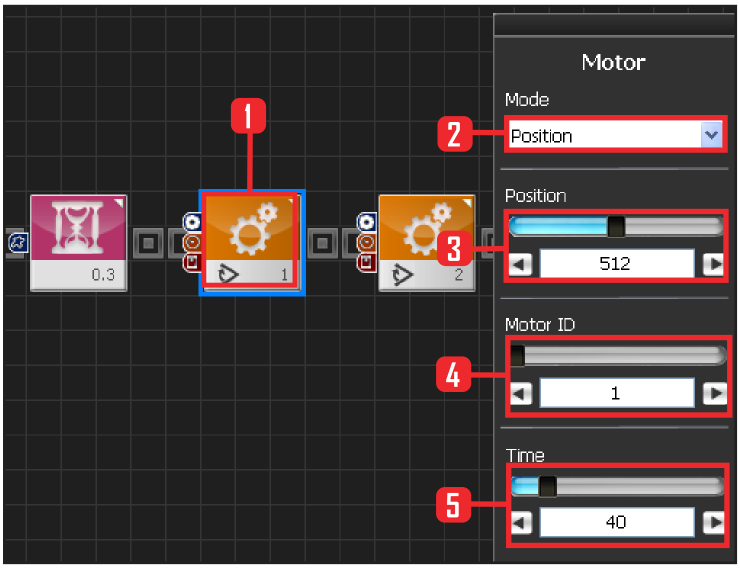
ウェーブを終わります。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：512に設定します。
Motor ID：4に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。


Delay
Time: 0.3

35 Delay　遅延させる

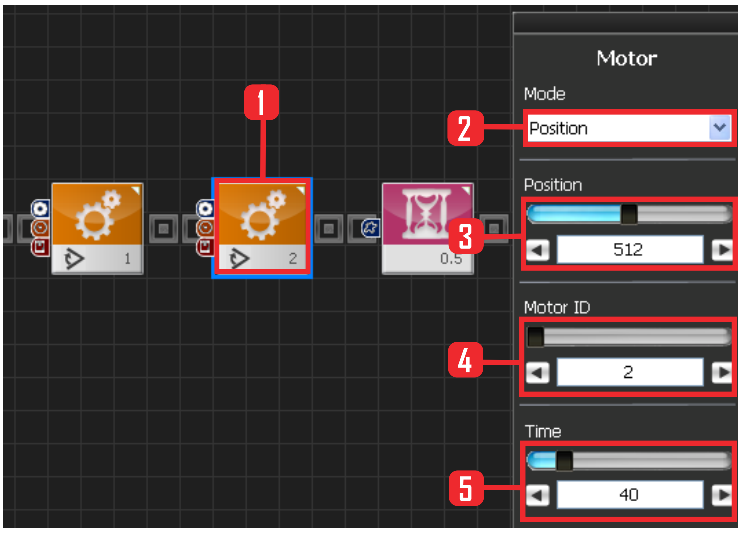
0.3秒間待ちます。
踊りのモーション部分なので、Delay　Timeを短く設定します。


Motor
Mode: Position
Position: 512
Motor ID: 1
Time: 40

36 モーター1番（右腕の上）の設定

8段階：ウェーブ5段階
ロボットの両腕を広げた初期姿勢に復帰させます。

Motion > Motorを選択します。
Mode：Positionを選択します。
Position：512に設定します。
Motor ID：1に設定します。
Time：40に設定します。約0.448秒間かけて設定角度に移動します。



Motor

Mode
Position

Position
512

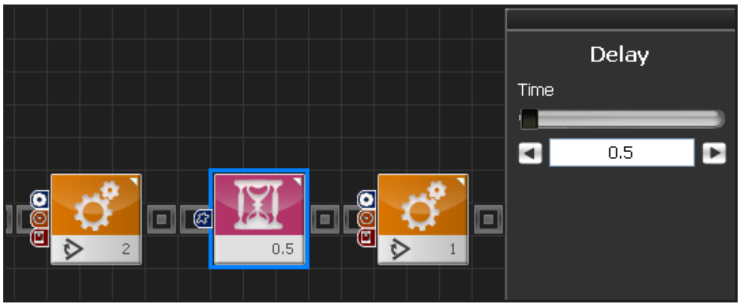
Motor ID
2

Time
40

37 モーター2番（右腕の下）の設定

ロボットの両腕を広げた初期姿勢に復帰させます。

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 512に設定します。
Motor ID : 2に設定します。
Time : 40に設定します。約0.448秒間かけて設定角度に移動します。

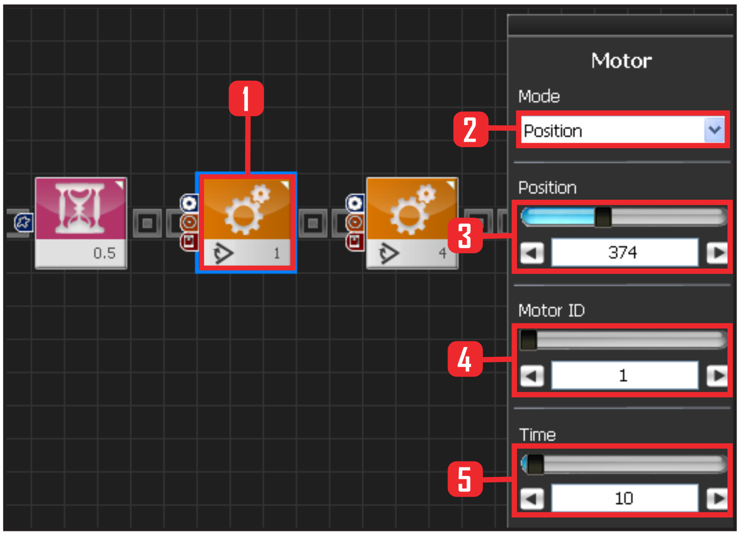


Delay

Time
0.5

38 Delay　遅延させる

0.5秒間待ちます。



Motor

Mode
Position

Position
374

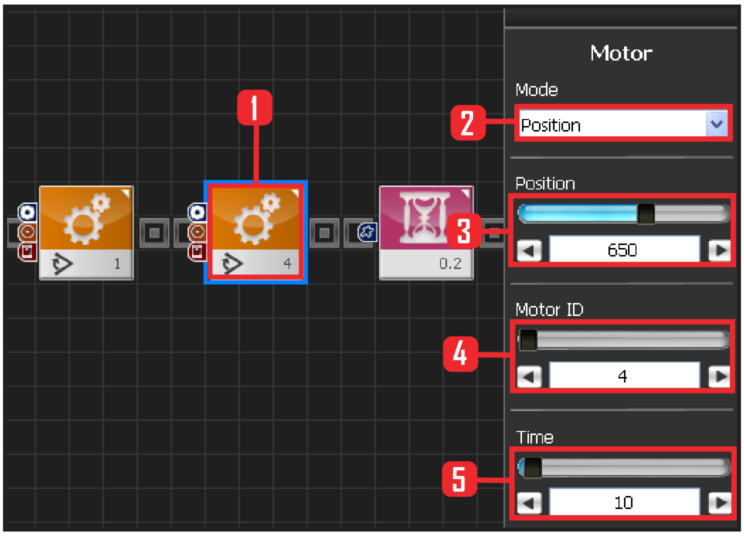
Motor ID
1

Time
10

39 モーター1番（右腕の上）の設定

9段階：45度の角度に腕を下げる。
直立不動の姿勢に戻ります。まず、45度の角度に移動します。

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 374に設定します。
Motor ID : 1に設定します。
Time : 10に設定します。

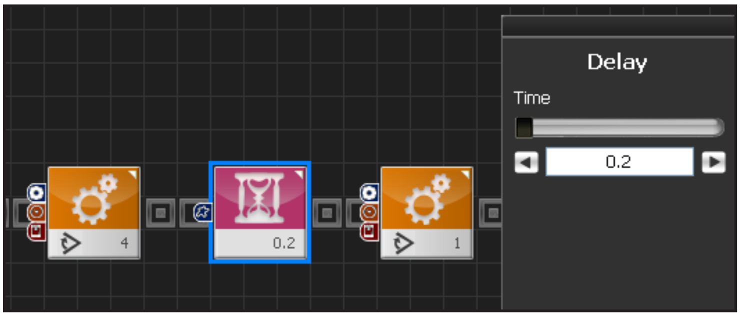


Motor
Mode: Position
Position: 650
Motor ID: 4
Time: 10

40 モーター4番（左腕の上）の設定

直立不動の姿勢に戻ります。まず、45度の角度に移動します。

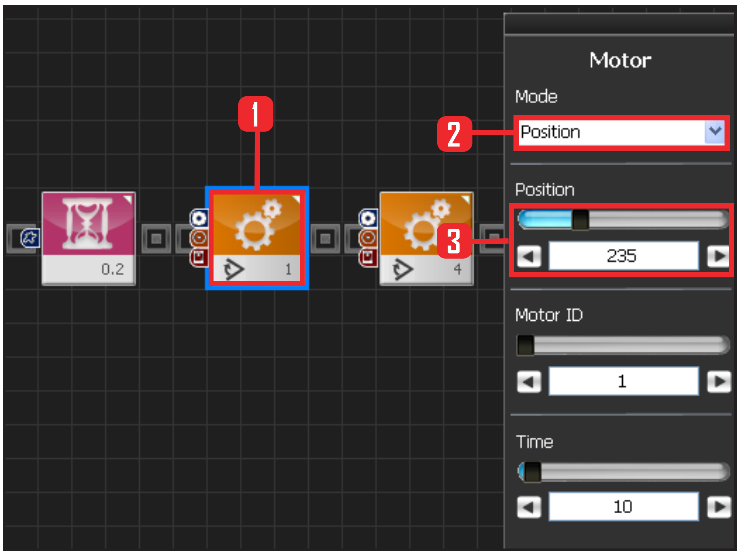
Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 650に設定します。
Motor ID : 4に設定します。
Time : 10に設定します。



Delay
Time: 0.2

41 Delay 遅延させる

0.2秒間待ちます。



Motor
Mode: Position
Position: 235
Motor ID: 1
Time: 10

42 モーター1番（右腕の上）の設定

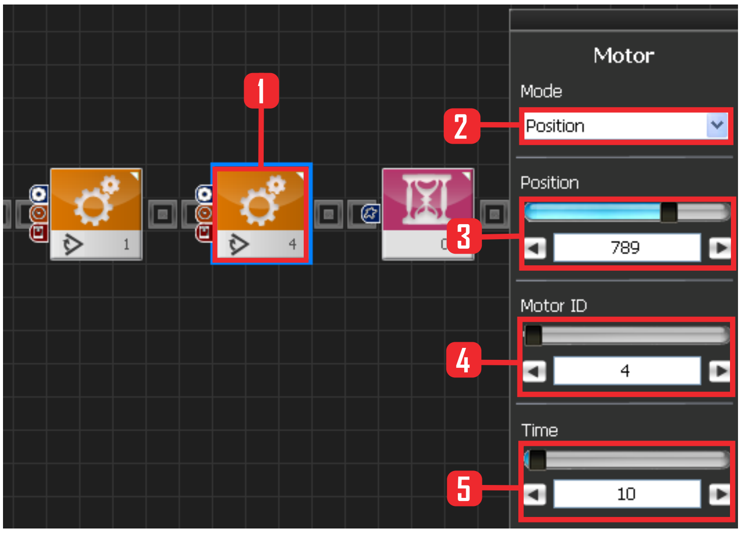
10段階 : 踊りの完了
直立不動の姿勢に戻ります

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 235に設定します。
Motor ID : 1に設定します。
Time : 10に設定します。約0.448秒間かけて設定角度に移動します。

43 モーター4番（右腕の上）の設定

直立不動の姿勢に戻ります

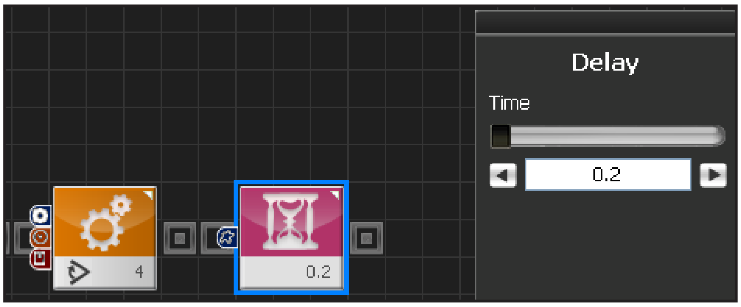
Motion > Motorを選択します。
Mode：Positionを選択します。
Position：789に設定します。
Motor ID：4に設定します。
Time：10に設定します。



Motor
Mode: Position
Position: 789
Motor ID: 4
Time: 10

44 Delay　遅延させる

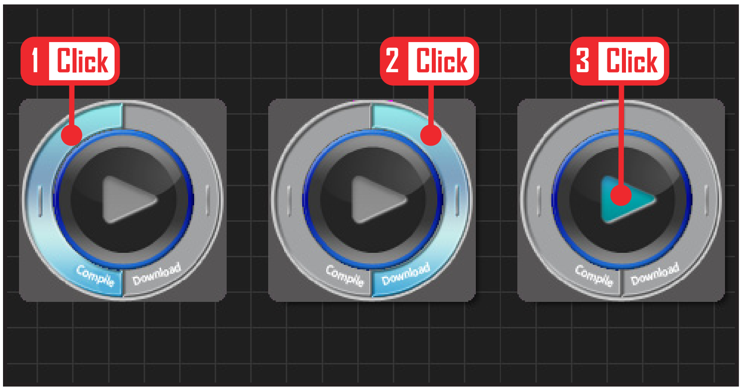
0.2秒間待ちます。

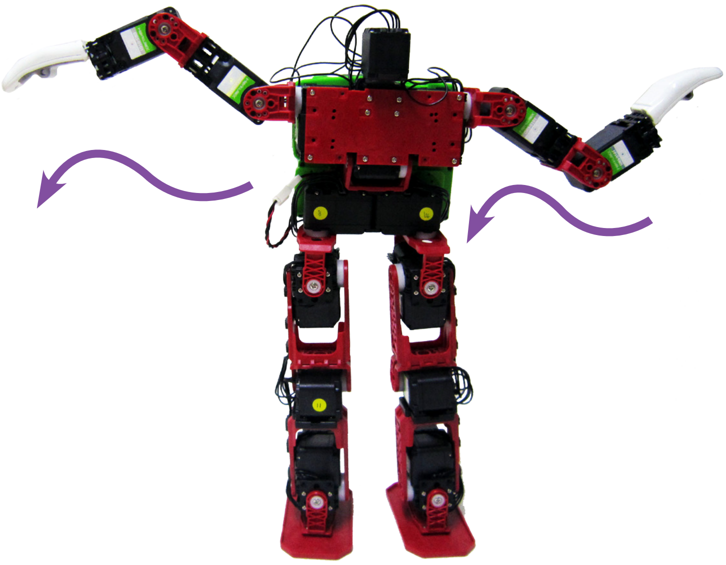


Delay
Time: 0.2

45 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download



46 ロボット動作

ロボットが左腕からウェーブをします。

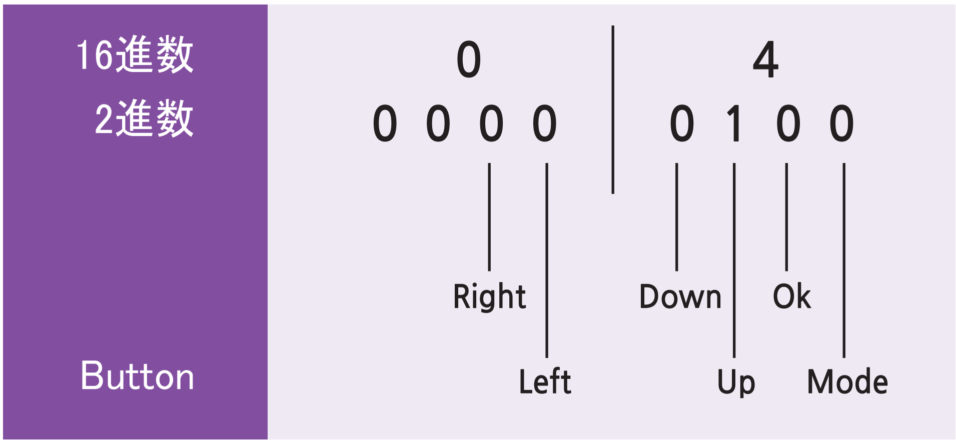
Button、LEDの例

例の説明

DRCコントローラーのButtonを使用して、LEDをオン/オフするプログラミングです。

Button、LEDのプログラミングをするためにまずButtonでLEDを点灯する方法を知っておく必要があります。

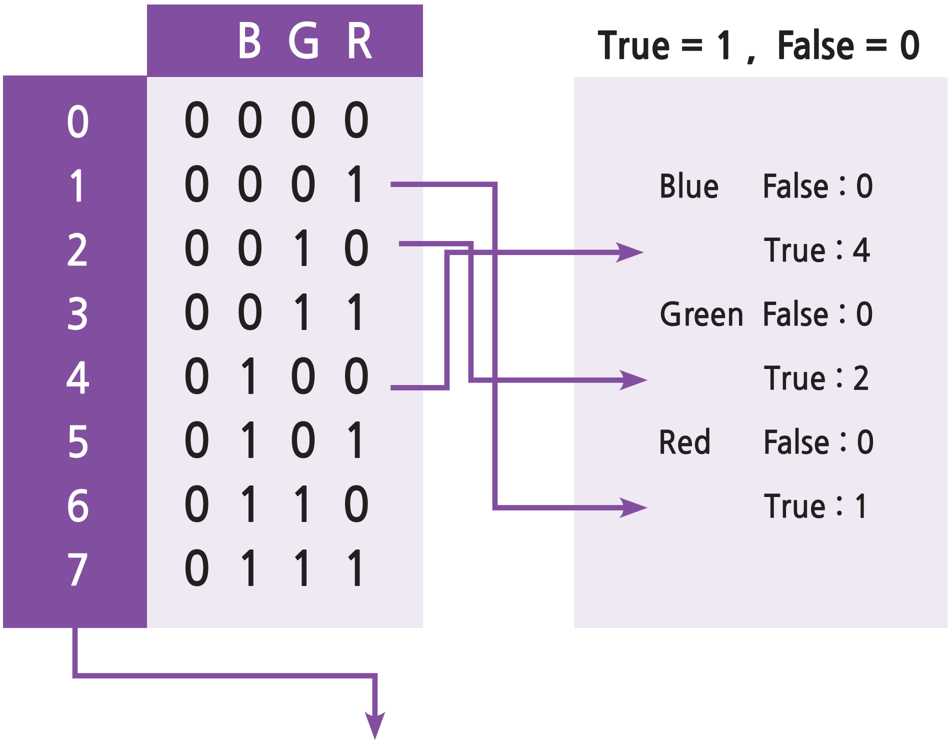
$16 = 2^4$



EX)	Right	2	0	h
		0 0 1 0	0 0 0 0	
	Down	0	8	h
		0 0 0 0	1 0 0 0	

Button

DRCには6つのボタンがあり、ボタンが押された状態を1Byteで表示します。　1Byteは8Bitで構成されているので、1Byteには8個の1と0を保存することができます。　DRCのボタン6個が押された状態（1）と押されていない状態（0）を表示するためには6Bitが必要です。上図に示すように、各ボタンはBit一つでマッチングしています。
ボタンが押された状態は1と0で表示され、これはButtonのモジュールの右下に16進数で表示されます。　Rightボタンが押された場合、ボタンの値は00100000となり、これを16進数に変更すると20hです（hは16進数という表示です）。　Downボタンが押された場合、ボタンの値は00001000となり、これを16進数に変更すると08hです。もう少し複雑なものとしてUp+Downボタンが押された場合、ボタンの値は00001100となり、これを16進数に変更すると0Chになります。



		B	G	R
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

True = 1 , False = 0

Blue	False : 0
	True : 4
Green	False : 0
	True : 2
Red	False : 0
	True : 1

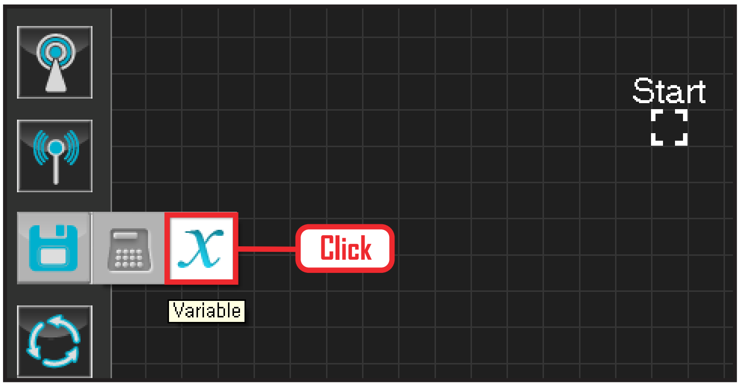
led( )

led(4 X Blue + 2 X Green + 1 X Red)

LED

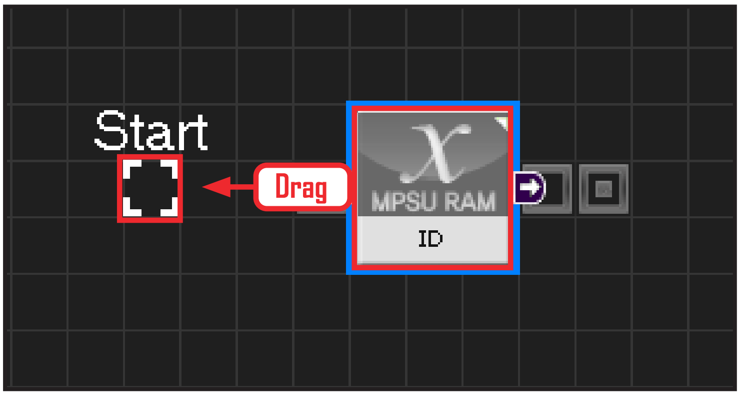
DRCには7つのLEDがありますが、Taskモードで制御可能なのは3つです。 Red、Green、BlueのLED3つのオン状態とオフ状態を表示するために3Bitが必要です。上図に示すように、各LEDはByteの最下位BitからRed、Green、Blueの順にマッチングしています。このLED値をLEDモジュールの入力として入力すると、該当するLEDがオンになります。例えば、LEDモジュールの入力値に2を入力した場合、2は2進数で表示すると、00000010なのでGreen LEDがオンになります。同じ原理で、入力値0(00000000)はすべてのLEDをオフにします。入力値7(00000111)は、すべてのLEDをオンにします。
Blueは2進数では4の位置に相当し、Greenは2の位置、Redは1の位置に対応しています。もし、Blue、Green、Redという名前の変数があり、これらの値(true、false)に基づいて、各LEDをオン/オフするかどうかを決める場合、LEDモジュールの入力値に4x Blue+2 x Green+1 x Redを入れると、格変数名のLEDを制御することができます。
例えば、Blue、GreenがtrueでRedがfalseの時、4 x Blue+2 x Green+1 x Red=6になります。6は2進数で表示すると、00000110ですので、これをLEDモジュールの入力に入力するとGreen、Blue LEDがオンになります。

上記の背景知識に基づいてButtonとLEDのプログラミングをします。


Start
Click
Variable

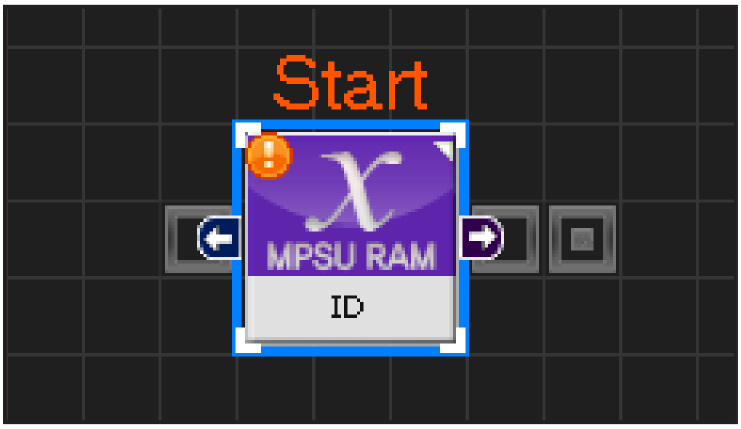
01 変数の指定

Data > Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせせます。

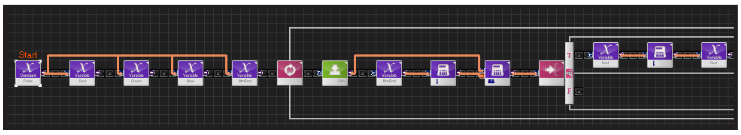

Start
MPSU RAM
ID

03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。
これはプログラミングが開始されたという意味です。

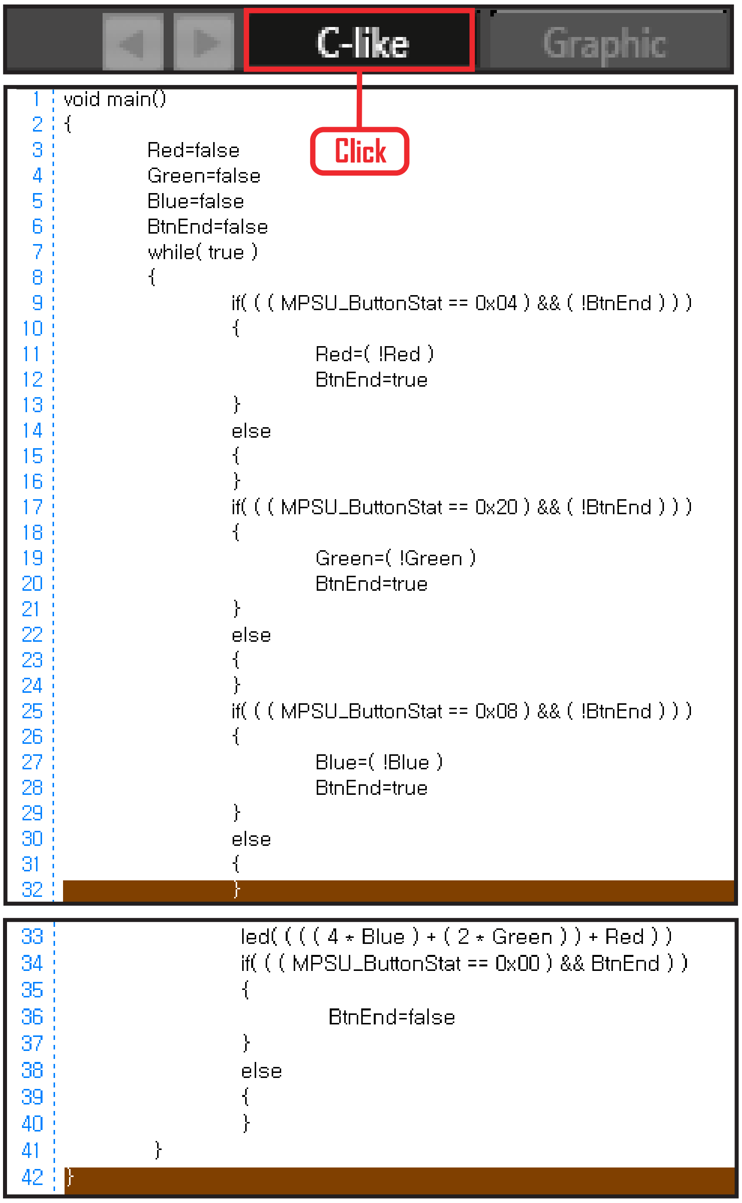
04 完全プログラミング

ボタンとLEDを利用した全体のプログラムです。

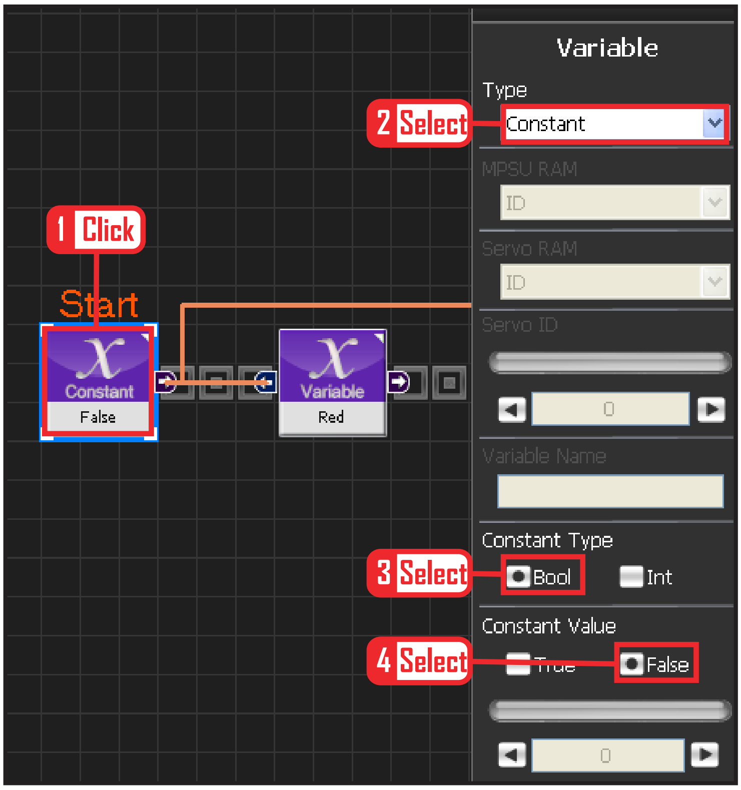


05 C-Like

右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
ボタンとLEDを利用して全体プログラムをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。



```
void main()
{
        Red=false
        Green=false
        Blue=false
        BtnEnd=false
        while( true )
        {
                if( ( ( MPSU_ButtonStat == 0x04 ) && ( !BtnEnd ) ) )
                {
                        Red=( !Red )
                        BtnEnd=true
                }
                else
                {
                }
                if( ( ( MPSU_ButtonStat == 0x20 ) && ( !BtnEnd ) ) )
                {
                        Green=( !Green )
                        BtnEnd=true
                }
                else
                {
                }
                if( ( ( MPSU_ButtonStat == 0x08 ) && ( !BtnEnd ) ) )
                {
                        Blue=( !Blue )
                        BtnEnd=true
                }
                else
                {
                }
                led( ( ( ( 4 * Blue ) + ( 2 * Green ) ) + Red ) )
                if( ( ( MPSU_ButtonStat == 0x00 ) && BtnEnd ) )
                {
                        BtnEnd=false
                }
                else
                {
                }
        }
}
```

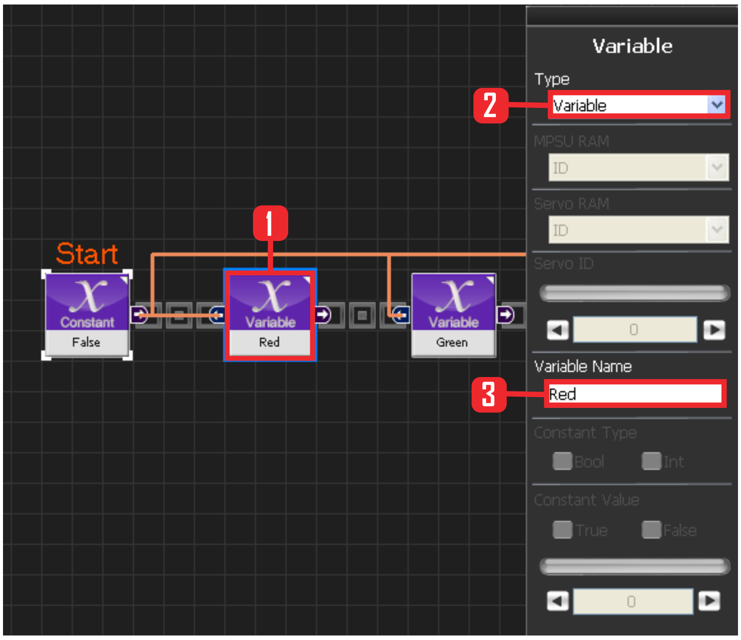

1 Click
Start
Constant
False
Variable
Red
Variable
2 Select
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
3 Select
Bool
Int
Constant Value
4 Select
True
False
0

06 Falseに初期化

すべてのLEDをFalse、オフの状態に初期化します。

Data> Variableモジュールを選択します。
Type:Contantを選択します。
Constant Typeは、Boolに設定します。BoolはTrueとFalseを表すデータ型です。
Constant Value:Falseを選択します

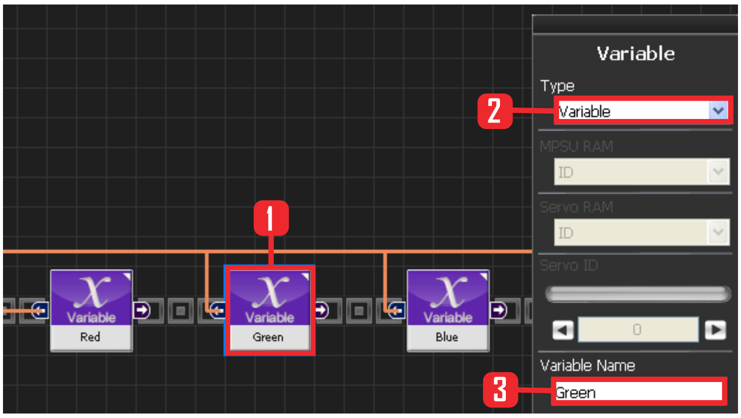
その後、コネクタを使用してFalseをそれぞれの変数に接続します。


Start
1
Constant
False
Variable
Red
Variable
Green
Variable
Type
2
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
3
Red
Constant Type
Bool
Int
Constant Value
True
False
0

07 Red 変数

Data > Variableを選択します。
Type:Variableを選択します。
Variable Name:Redを入力します。

赤LEDはFalseの場合はオン、Trueの場合はオフになります。



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

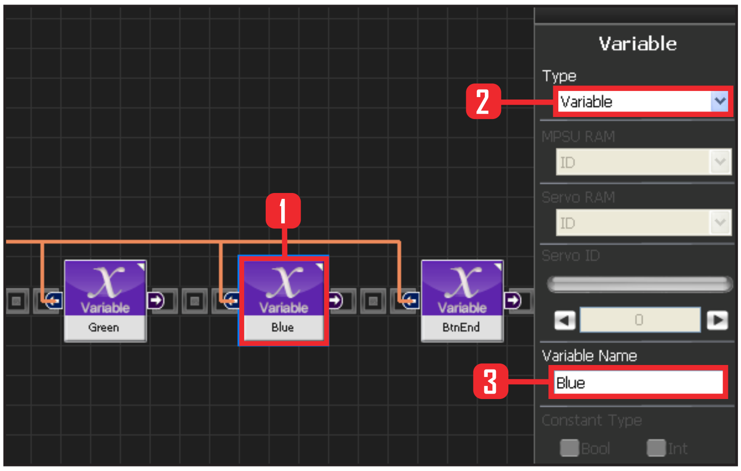
Variable Name: Green

1
2
3
Variable
Red
Variable
Green
Variable
Blue

08 Green 変数

Data > Variableを選択します。
Type : Variableを選択します。
Variable Name : Greenを入力します。

緑LEDはFalseの場合はオン、Trueの場合はオフになります。



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: Blue

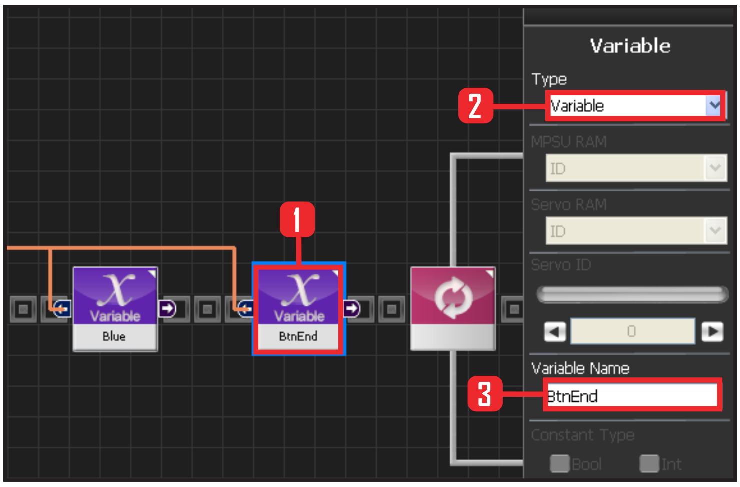
Constant Type: ☐ Bool ☐ Int

1
2
3
Variable
Green
Variable
Blue
Variable
BtnEnd

09 Blue 変数

Data > Variableを選択します。
Type : Variableを選択します。
Variable Name : Blueを入力します。

青LEDはFalseの場合はオン、Trueの場合はオフになります。



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: BtnEnd

Constant Type: ☐ Bool ☐ Int

1
2
3
Variable
Blue
Variable
BtnEnd

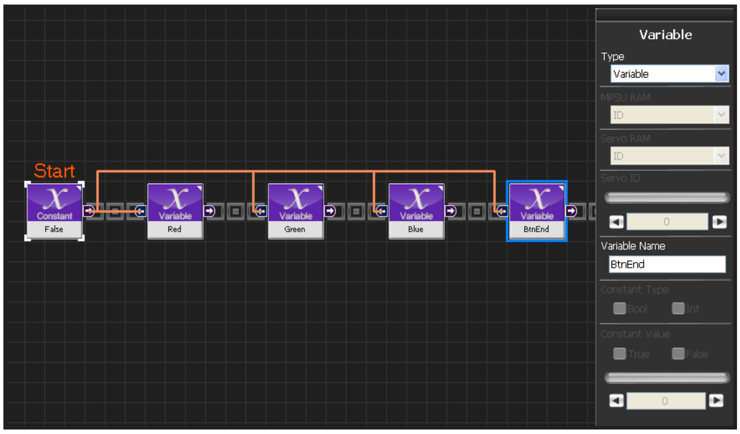
10 BtnEnd 変数

ボタンが押されていない時はFalseのままにしてボタンが押された後、目的の作業が終わる瞬間にFalse -> True に変わる変数です。
また、ボタンを離す瞬間にTrueからFalseに変わる変数です。

Data> Variableを選択します。
Type : Variableを選択します。
Variable Name : BtnEndを入力します。

11 変数の宣言

Red、Green、Blue、BtnEnd 4つのすべての初期値をFalseに設定します。

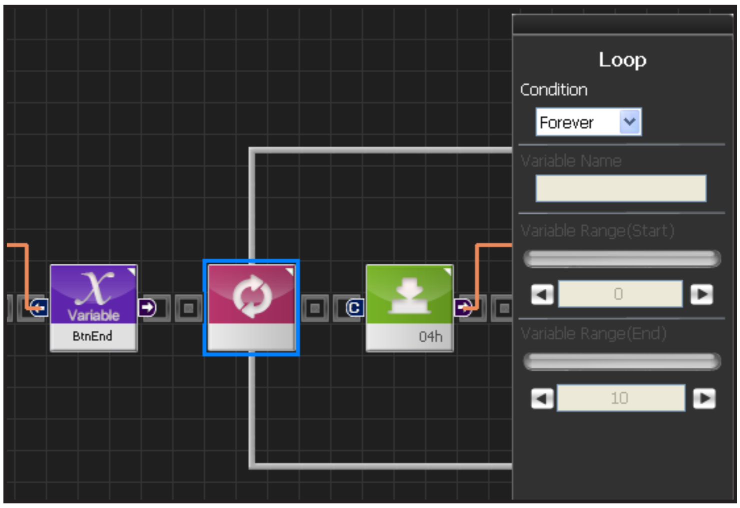

Start
Constant
False
Variable
Red
Variable
Green
Variable
Blue
Variable
BtnEnd
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Constant Type
Bool
Int
Constant Value
True
False
0

Type: Variable

Variable Name: BtnEnd

12 Loop

Forever 無限に繰り返します。


Variable
BtnEnd
04h
Loop
Variable Name
Variable Range(Start)
0
Variable Range(End)
10

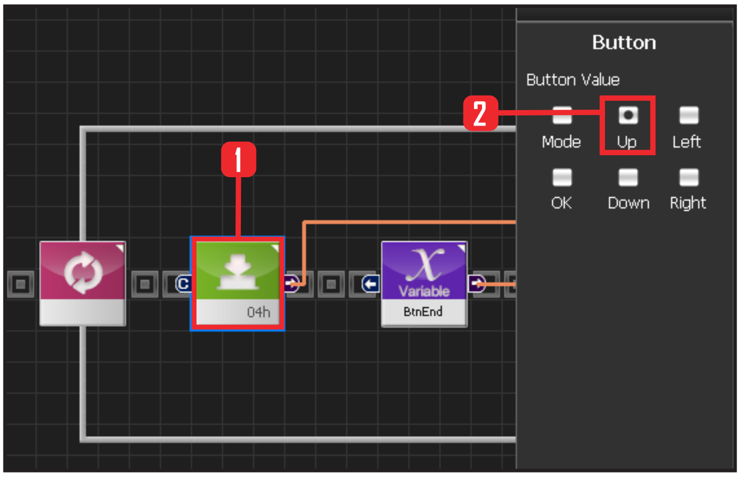
Condition: Forever

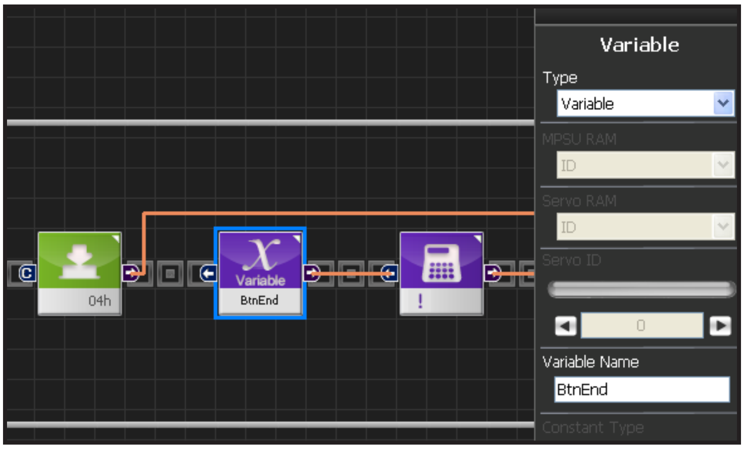
13 Upボタン

ボタンモジュールを作成します。このモジュールは選択したボタンが押された場合はTrueになり、それ以外はFalseになります。Up Buttonを選択すると、Up Buttonが押された場合はtrue、それ以外にfalseになります。

Communication > Buttonモジュールを選択します。
Button ValueのUpを選択します。

16進数で04hなので、モジュールの画像に04hが表記されます。


1
2
04h
Variable
BtnEnd
Button
Button Value
Mode
Up
Left
OK
Down
Right

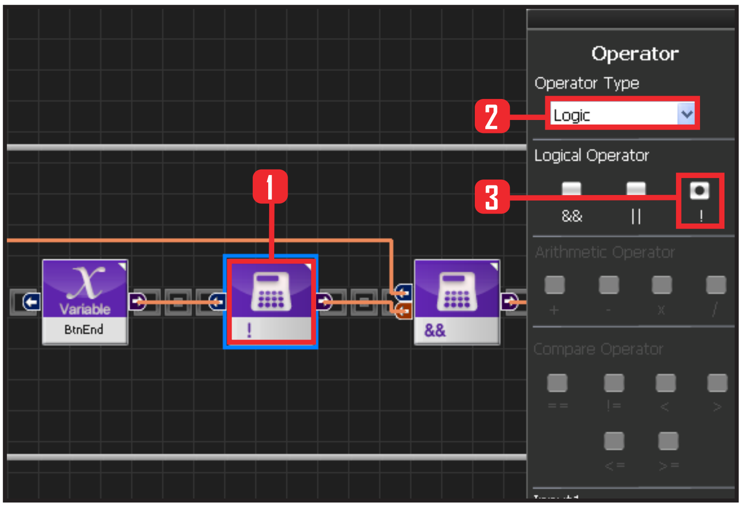

C
04h
Variable
BtnEnd
!
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Constant Type

Variable
Type: Variable
Variable Name: BtnEnd

14 BtnEnd

BtnEnd値はFalseに初期化されています。後ろにnot演算子がついてTrueになります。

前のbtnEnd変数をコピーして貼り付けます。

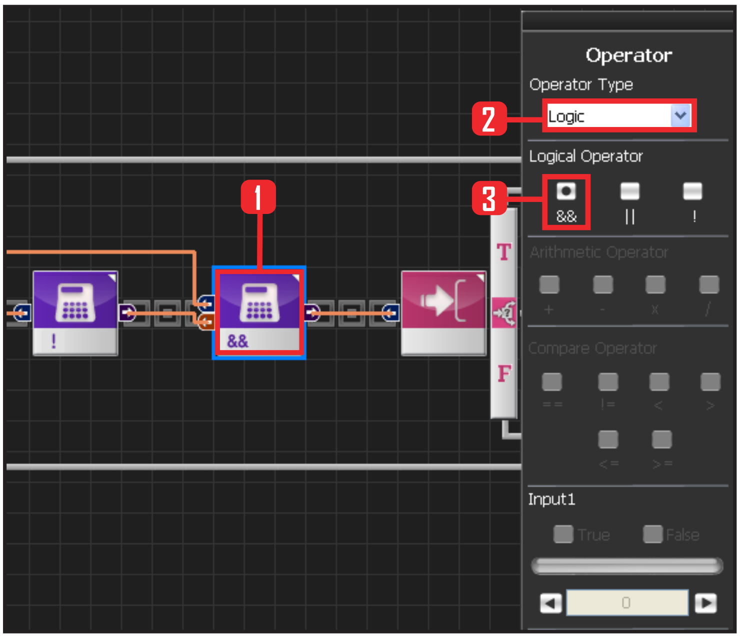

1
2
3
Variable
BtnEnd
!
&&
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>
<=
>=

Operator
Operator Type: Logic
Logical Operator: ○ && ○ || ● !

15 ！演算

！演算を適用してBtnEndの値を反転します。

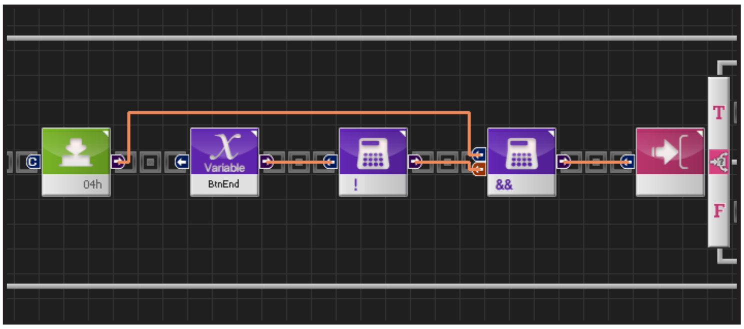
Data＞Operatorモジュールを選択します。
Operartor Type：Logicを選択します。
Logical Operator：！を選択します。


1
2
3
!
&&
T
F
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>
<=
>=
True
False
0

Operator
Operator Type: Logic
Logical Operator: ● && ○ || ○ !
Input1

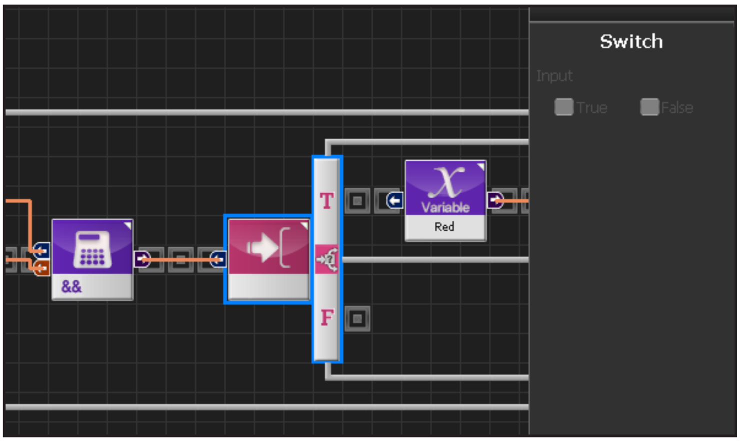
16 And 演算

Upボタンが押され、BtnEndがfalse（！適用して、True）の時、Trueになり、後の条件文を実行します。
Data＞Operatorモジュールを選択します。
Operartor Type：Logicを選択します。
Logical Operator：＆＆を選択します。


C
04h
Variable
BtnEnd
!
&&
T
F

17 Upボタンが押された時

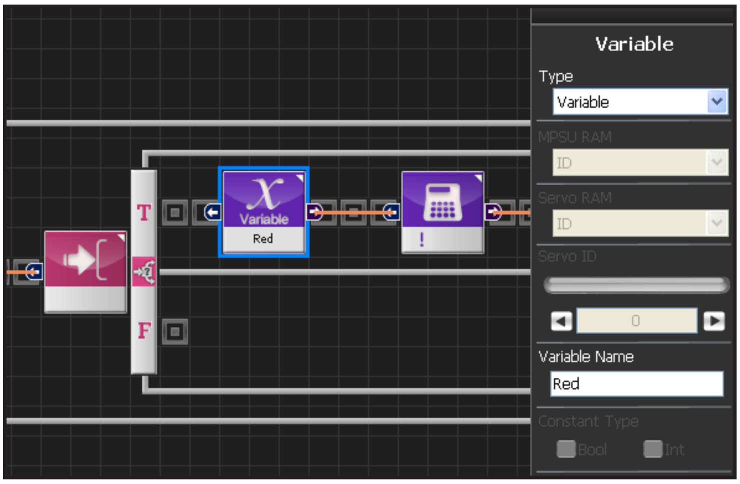
Upボタンが押され、BtnEndがfalseの時、後ろの条件を実行します。


&&
T
F
Variable
Red
Switch
Input

☐True ☐False

18 If 分岐文

Trueのとき、上の部分を実行します。


T
F
Variable
Red
!
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
Constant Type

Type: Variable

0

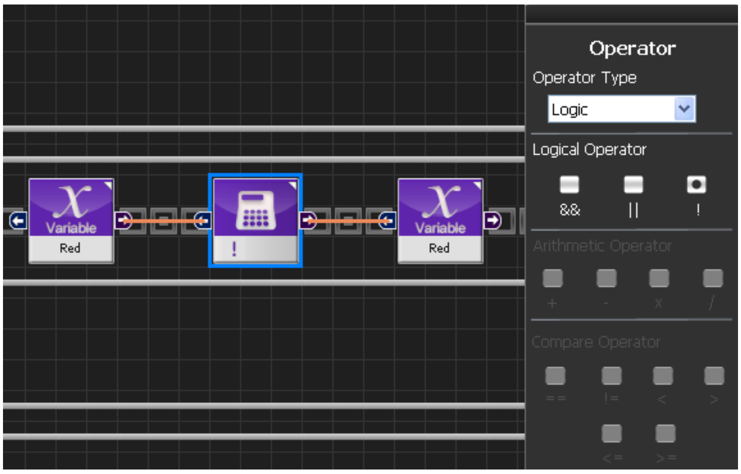
Variable Name: Red

☐Bool ☐Int

19 Red 出力

前のRed変数をコピーして貼り付けます。

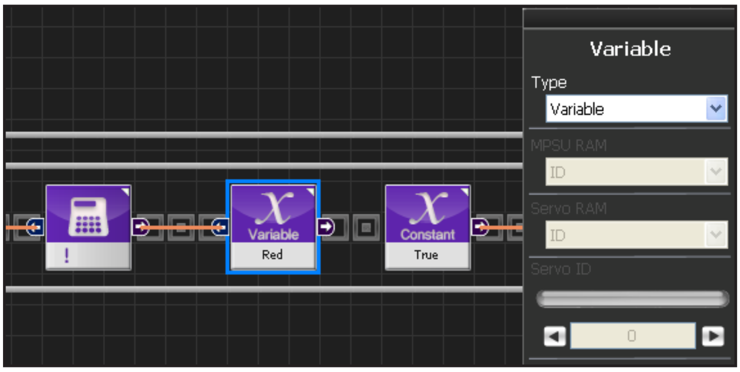
20 ！演算


Variable
Red
!
Variable
Red
Operator
Logical Operator
&&
||
!
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>
<=
>=

Operator Type: Logic

RedがTrueの場合はFalseになり、Falseの場合はTrueになります。

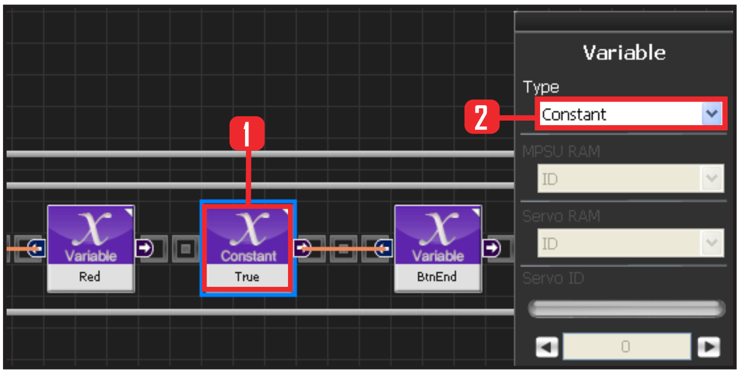
21 Red 入力


!
Variable
Red
Constant
True
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0

Type: Variable

前のRed変数の値がTrueの場合はFalseになり、Falseの場合はTrueに変わって保存されます。

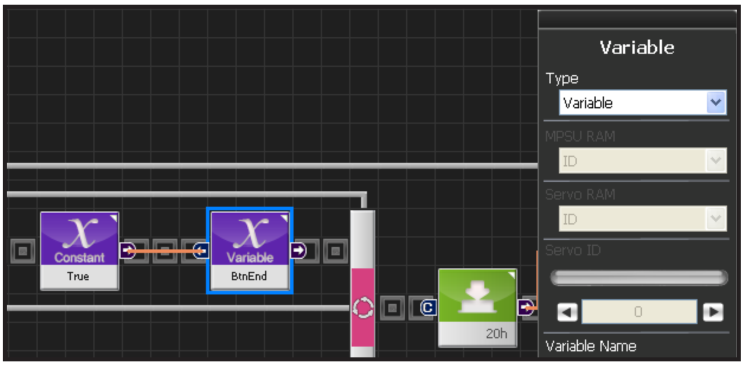
22 True 設定


Variable
Red
Constant
True
Variable
BtnEnd
1
2
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0

Type: Constant

ボタンが押され、関連する動作も終わったのでBtnEndをFalseからTrueに変更しなければなりません。

Data > Variableモジュールを選択します。
Type：Contantを選択します。
Constant Typeは、Boolに設定します。 BoolはTrueとFalseを表すデータ型です。
Constant Value：Trueを選択します

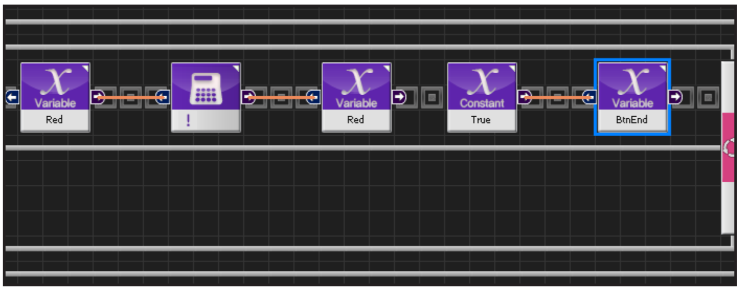

Constant
True
Variable
BtnEnd
20h
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name

Variable

Type: Variable

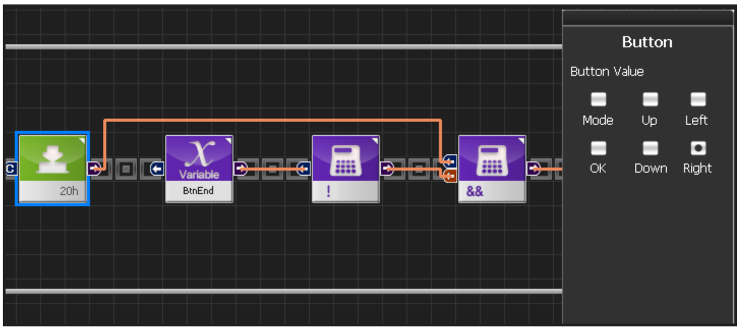
23 BtnEndをTrueに

True値がBtnEndに入力されます。
BtnEndがTrueになると、loop反復時にupボタンが押されていても条件を満たさないのでRed変数の値はこれ以上変わりません。


Variable
Red
!
Variable
Red
Constant
True
Variable
BtnEnd

24 Red LED

Upボタンを一度押すと、Redがオンになります。もう一度押すと、Redがオフになります。


20h
Variable
BtnEnd
!
&&

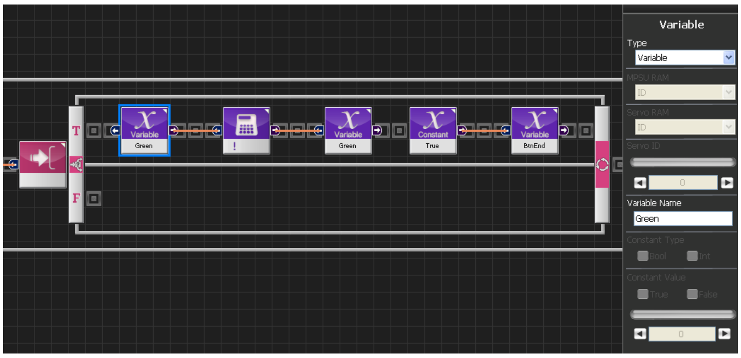
Button

Button Value

☐ Mode ☐ Up ☐ Left
☐ OK ☐ Down ☒ Right

25 Right ボタン

Rightボタンを押した場合です。

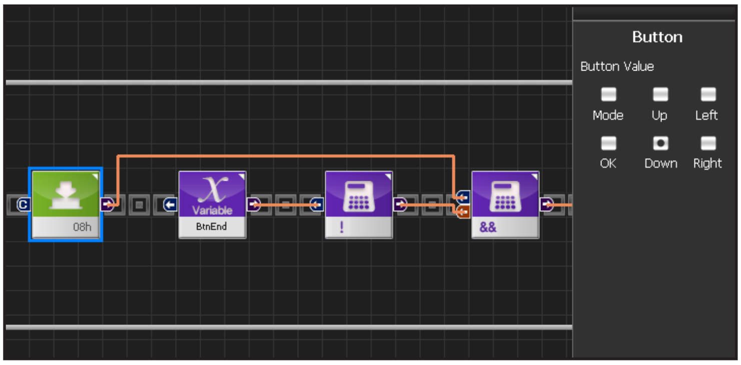

Variable
T
F
Variable
Green
!
Variable
Green
Constant
True
Variable
BtnEnd
MPSU RAM
Servo RAM
Servo ID
Constant Type
Bool
Int
Constant Value
True
False

Type: Variable

Variable Name: Green

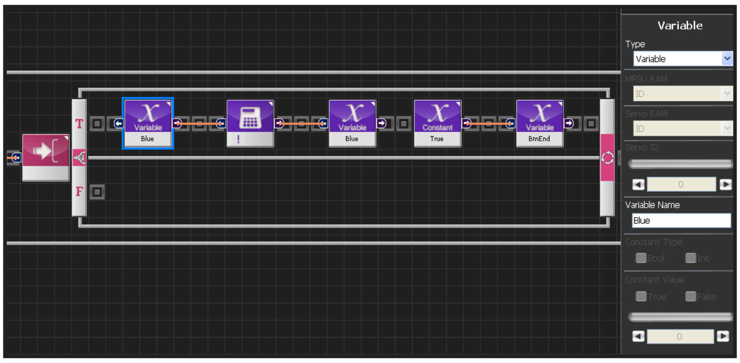
26 Green LED

Rightボタンを一回押すとGreenがオンになり、もう一度押すとGreenがオフになります。


Button
Button Value
Mode
Up
Left
OK
Down
Right
C
08h
Variable
BtnEnd
!
&&

27 Down ボタン

Downボタンを押した場合です。


Variable
T
F
Variable
Blue
!
Variable
Blue
Constant
True
Variable
BtnEnd
MPSU RAM
Servo RAM
Servo ID
Constant Type
Bool
Int
Constant Value
True
False

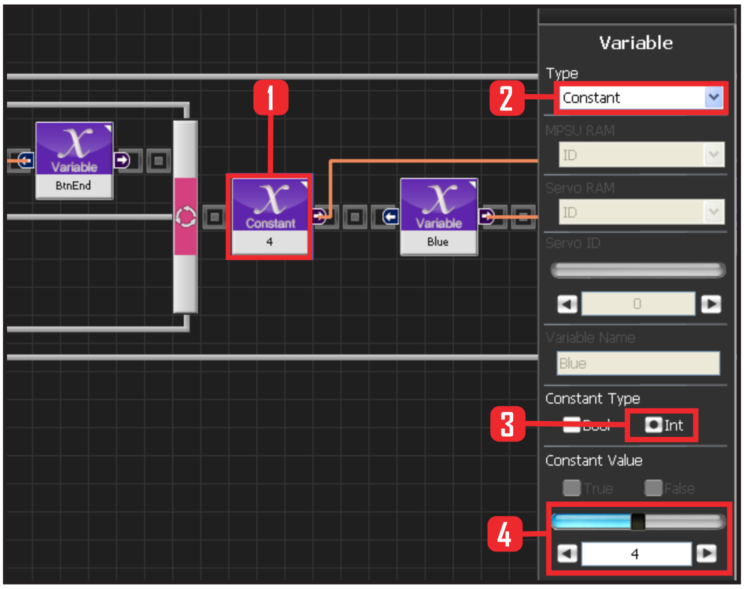
Type: Variable

Variable Name: Blue

28 Blue LED

Downボタンを一回押すとBlueがオンになり、もう一度押すとBlueがオフになります。

29 LED の値


Variable
BtnEnd
Constant
4
Variable
Blue
1
2
3
4
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Blue
Constant Type
Bool
Int
Constant Value
True
False
4

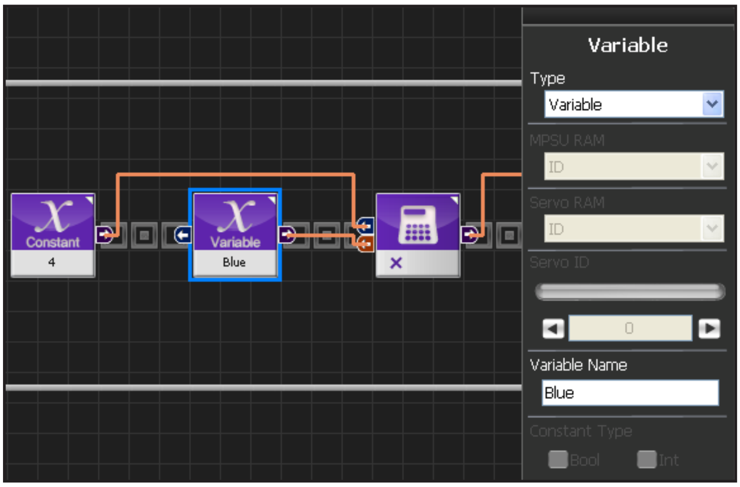
前述したようにLEDモジュールに値を入力すると、その値に応じてLEDがオンになります。その式は(4 x Blue+2 x Green+1 x Red)です。
上記の式をモジュールごとに接続して表現したのが左図です。

定数4を設定します。

(4 x Blue+2 x Green+1 x Red)

Data > Variableモジュールを選択します。
Type:Contantを選択します。
Constant Typeはintに設定します。
Constant Value:4に設定します。

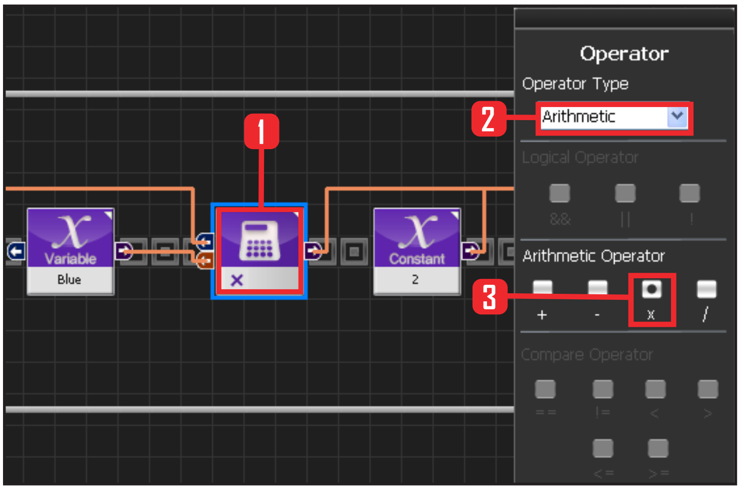
30 Blue


Constant
4
Variable
Blue
×
Variable
Type
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Blue
Constant Type
Bool
Int

(4 x Blue + 2 x Green + 1 x Red)

前のBlue変数をコピーします。

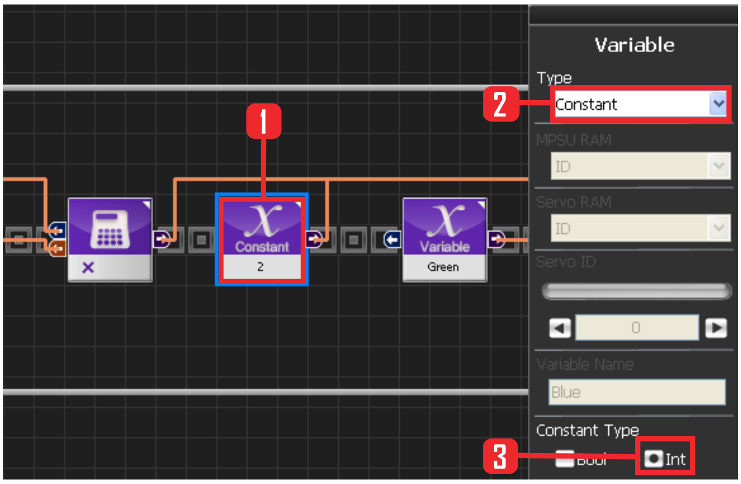
31 掛け算


Variable
Blue
×
Constant
2
1
2
3
Operator
Operator Type
Arithmetic
Logical Operator
&&
||
!
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>
<=
>=

(4 x Blue+2 x Green+1 x Red)

Data > Operatorモジュールを選択します。
Operartor Type:Arithmeticを選択します。
Arithmetic Operator:Xを選択します。

掛け算モジュールの2つの入力コネクタに定数4とBlue変数のモジュールをつなぎます。

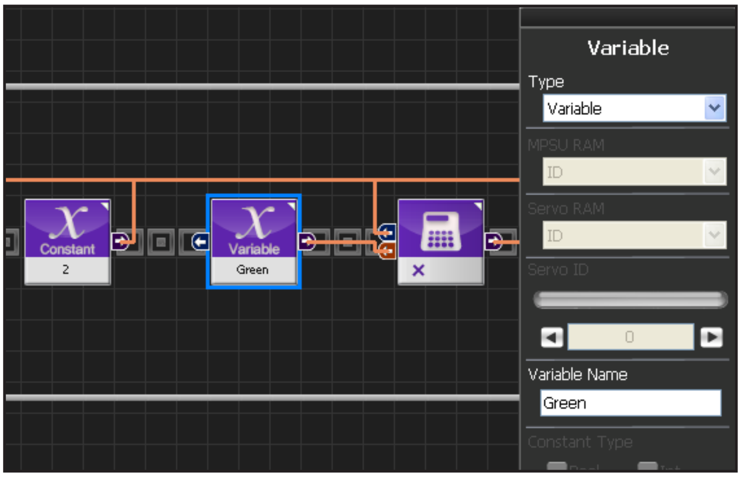

1
2
3
×
Constant
2
Variable
Green

Variable
Type: Constant
MPSU RAM: ID
Servo RAM: ID
Servo ID: 0
Variable Name: Blue
Constant Type: ○ Bool ● Int

32 定数2

（4 x Blue+2 x Green+1 x Red）

Data > Variableモジュールを選択します。
Type：Contantを選択します。
Constant TypeはIntに設定します。
Constant Value：2に設定します。

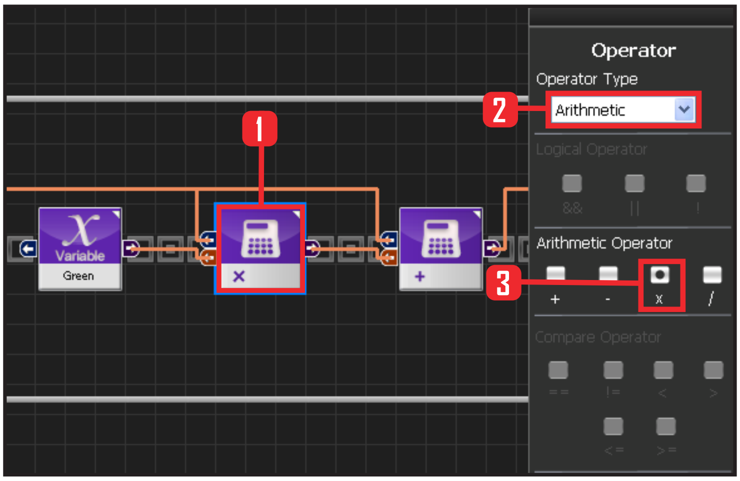

Constant
2
Variable
Green
×

Variable
Type: Variable
MPSU RAM: ID
Servo RAM: ID
Servo ID: 0
Variable Name: Green
Constant Type: ○ Bool ○ Int

33 Green

（ 4 x Blue + 2 x Green + 1 x Red ）

前のGreenの変数をコピーします。


1
2
3
Variable
Green
×
+

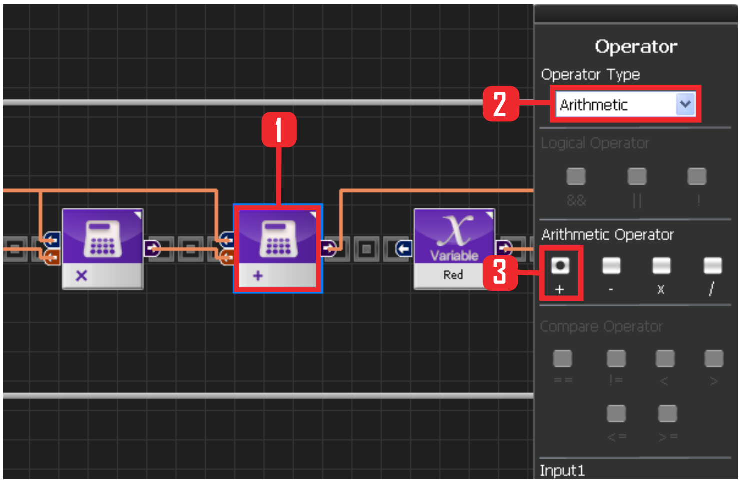
Operator
Operator Type: Arithmetic
Logical Operator: ○ && ○ || ○ !
Arithmetic Operator: ○ + ○ - ● x ○ /
Compare Operator: ○ == ○ != ○ < ○ > ○ <= ○ >=

34 掛け算

（4 x Blue+2 x Green+1 x Red）

Data > Operatorモジュールを選択します。
Operartor Type：Arithmeticを選択します。
Arithmetic Operator：Xを選択します。

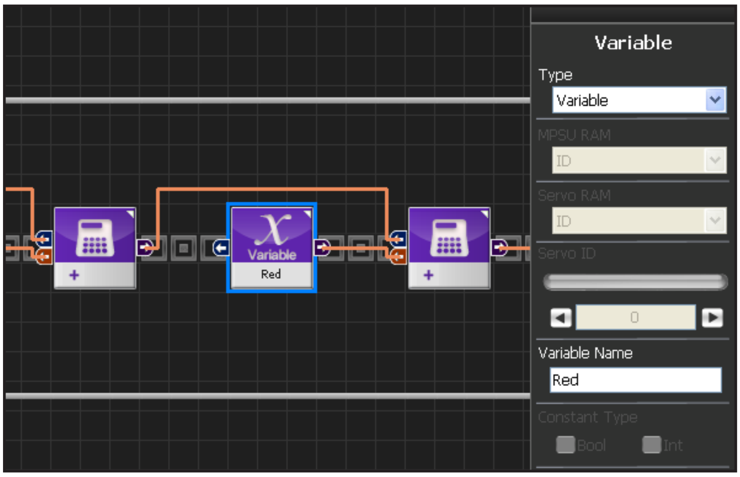
掛け算モジュールの2つの入力コネクタに定数2とGreenの変数のモジュールをつなぎます。


Operator
Operator Type
Arithmetic
Logical Operator
Arithmetic Operator
+
-
x
/
Compare Operator
Input1
Variable
Red
1
2
3

35 足し算

Data > Operatorモジュールを選択します。
Operartor Type：Arithmeticを選択します。
Arithmetic Operator：+を選択します。

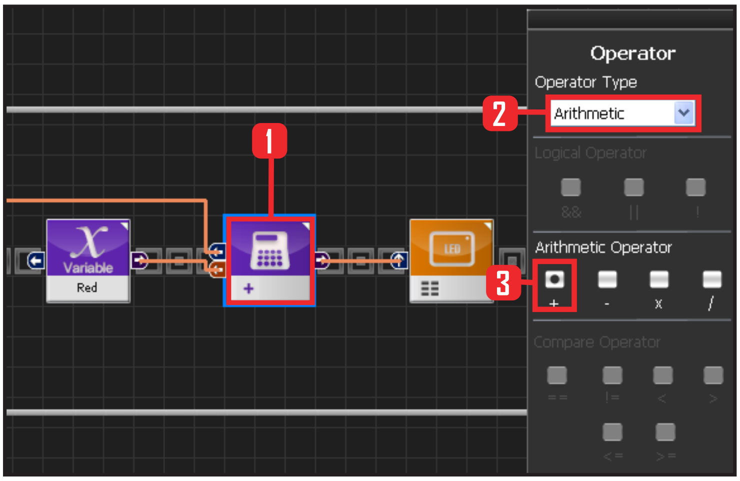
前の31番、34番からの掛け算モジュールの出力を足し算モジュールの2つの入力コネクタに接続します。


Variable
Type
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Red
Constant Type
Bool
Int
Variable
Red

36 Red

（4 x Blue + 2 x Green + 1 x Red）

前のRed変数をコピーします。

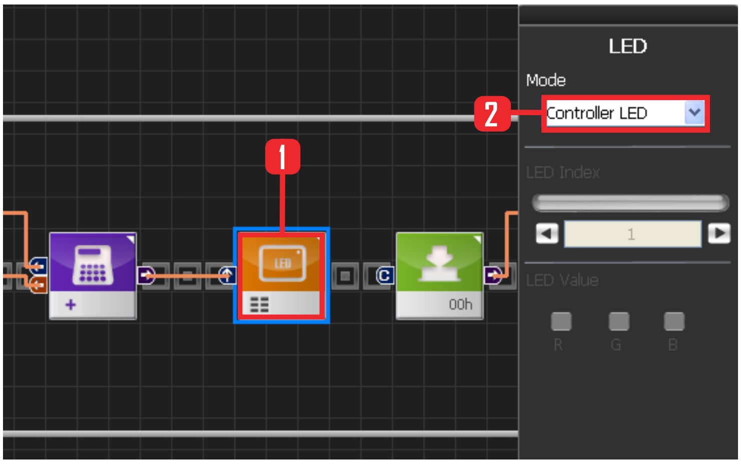

Operator
Operator Type
Arithmetic
Logical Operator
Arithmetic Operator
+
-
x
/
Compare Operator
Variable
Red
LED
1
2
3

37 足し算

（4 x Blue+2 x Green+1 x Red）

Data > Operatorモジュールを選択します。
Operartor Type：Arithmeticを選択します。
Arithmetic Operator：+を選択します。

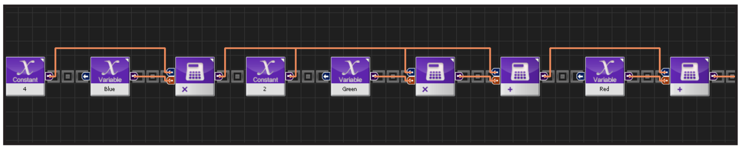
35番の足し算モジュールとRed変数モジュールの出力を足し算モジュールの2つの入力コネクタに接続します。


LED
Mode
Controller LED
LED Index
1
LED Value
R
G
B
00h
LED
+
1
2

38 LED

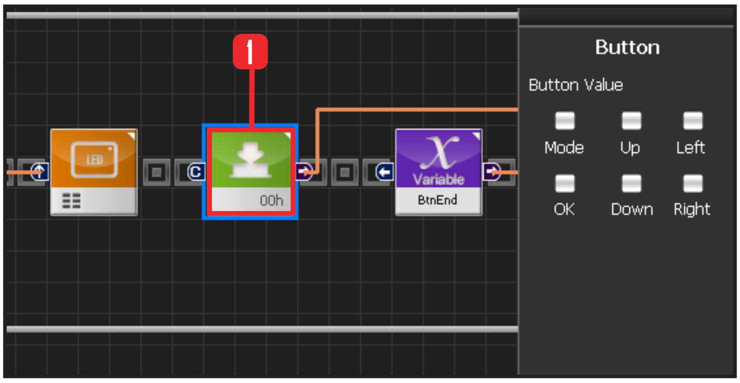
Motion > LEDモジュールを選択します。
Mode : Controller LEDを選択します。

前で演算した値をLED Valueに入れて、それぞれのLEDがオンになります。.


Constant
4
Variable
Blue
×
Constant
2
Variable
Green
×
+
Variable
Red
+

39 LED値を生成する演算

(4 x Blue + 2 x Green + 1 x Red)
をモジュールとして表現したものです。

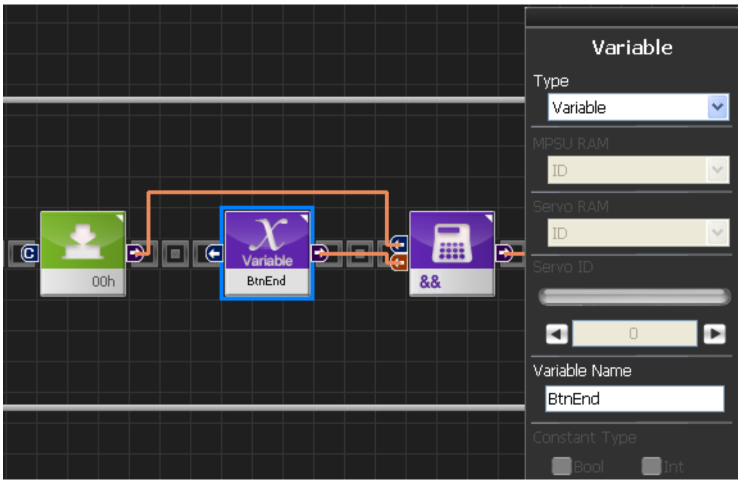

Button
Button Value
Mode
Up
Left
OK
Down
Right
00h
Variable
BtnEnd
LED
1

40 ボタンが押されていない状態

ボタンが押され、目的の動作を終えた後、BtnEnd変数はTrueに変更されます。 BtnEnd変数がTrueになると、ボタンに応じた動作をしないため一度押すと必要な動作を一度だけ実行であるようにすることができます。ボタンを離すと、BtnEndをFalseに初期化する必要があります。ここからは初期化過程をプログラミングします。

Motion > Buttonモジュールを選択します。
Button Value : 何も選択しません。

ボタンが押されていない状態を意味します。


Variable
00h
Variable
BtnEnd
&&
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Constant Type
Bool
Int

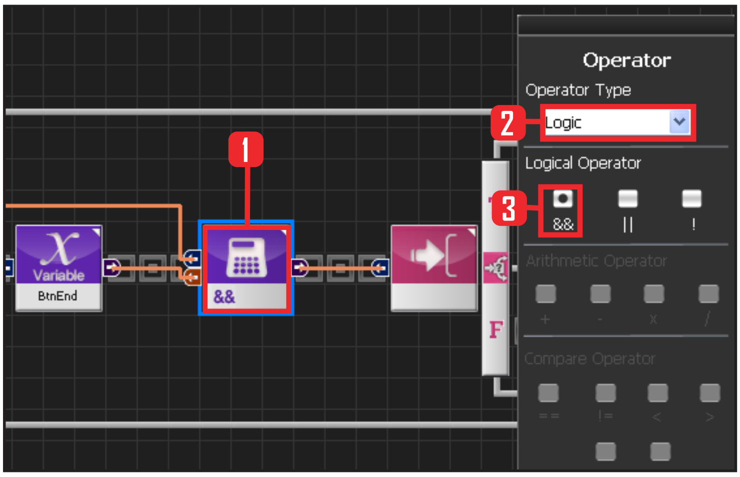
Type: Variable

Variable Name: BtnEnd

41 BtnEndがTrue

BtnEndがTrueの場合です。

前のBtnEnd変数をコピーします。


Operator
1
2
3
Variable
BtnEnd
&&
T
F
Logical Operator
&&
||
!
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>

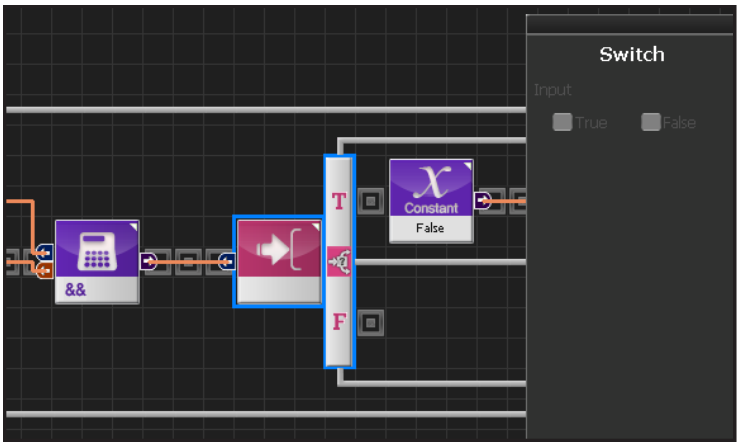
Operator Type: Logic

Logical Operator: ● &&

42 && 演算子

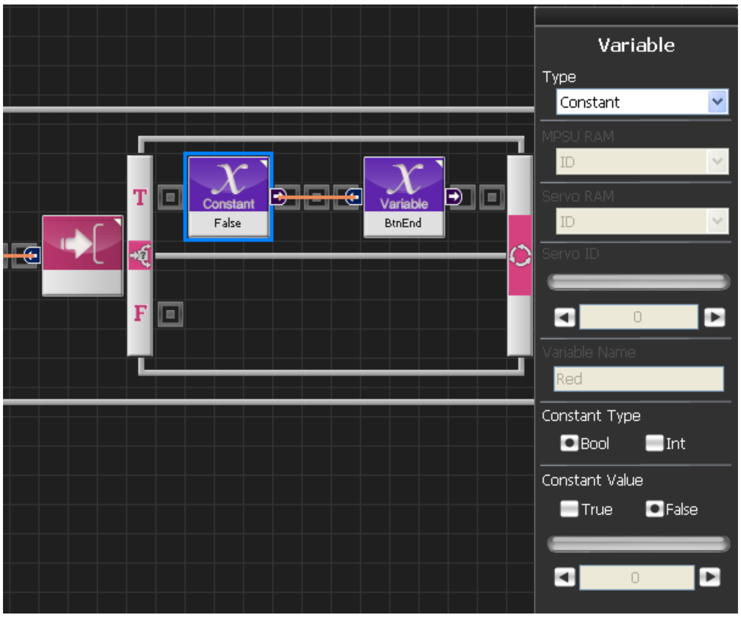
ボタンが押されていない状態とBtnEndがTrueである状態を同時に満足する状態はボタンを離した状態であることを意味します。

Data > Operatorモジュールを選択します。
Operartor Type : Logicを選択します。
Logical Operator : & &を選択します。


Switch
Input
True
False
&&
T
F
Constant
False

43 If 分岐文

ボタンを押して離すた状態がTrueの場合、実行されます。


T
F
Constant
False
Variable
BtnEnd

Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: Red

Constant Type: ● Bool ☐ Int

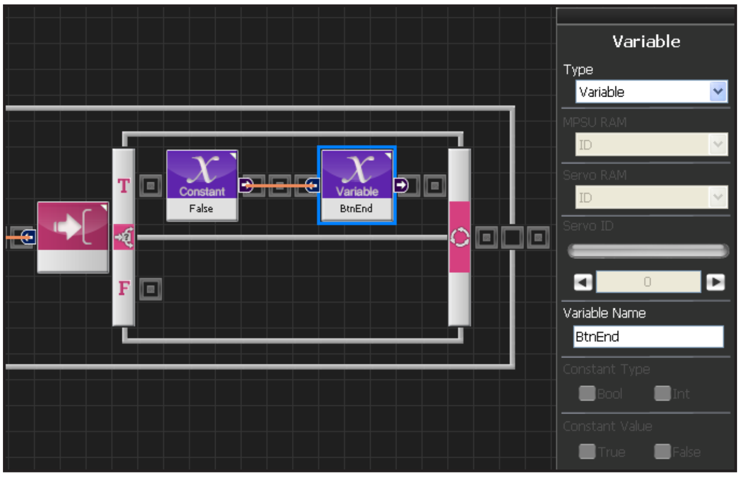
Constant Value: ☐ True ● False

0

44 False 値

BtnEndをTrueからFalseにします。

Data > Variableモジュールを選択します。
Type : Contantを選択します。
Constant　TypeはBoolに設定します。BoolはTrueとFalseを表すデータ型です。
Constant Value : Falseを選択します。


T
F
Constant
False
Variable
BtnEnd

Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name: BtnEnd

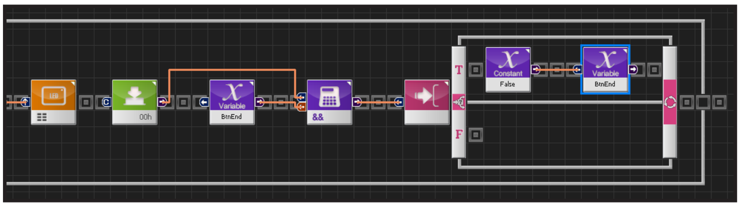
Constant Type: ☐ Bool ☐ Int

Constant Value: ☐ True ☐ False

45 BtnEndをFalseに

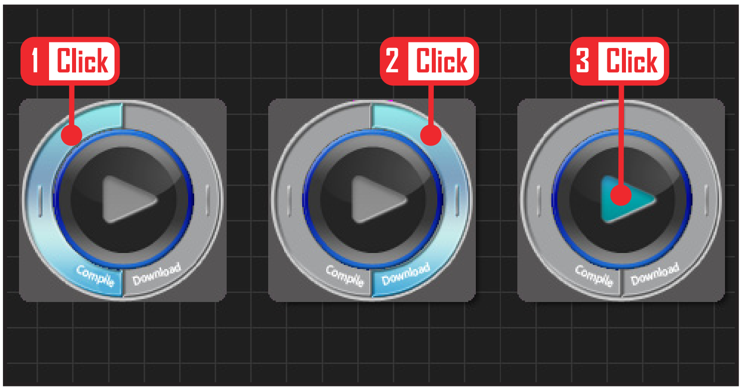
BtnEndにFalse値を入力します。

前のBtnEnd変数をコピーします。


00h
Variable
BtnEnd
&&
T
F
Constant
False
Variable
BtnEnd

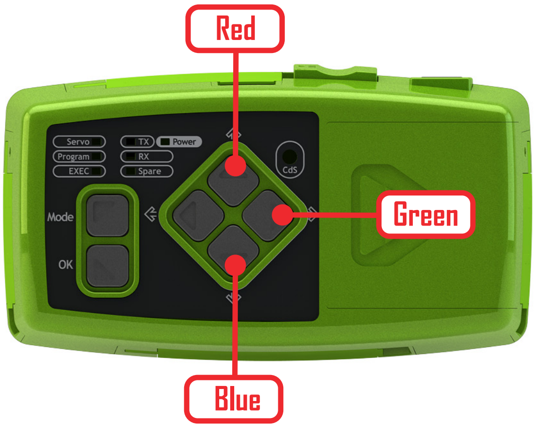
46 ボタンを離した時の初期化

ボタンを離した状態でBtnEndをFalseに初期化します。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

47 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。


Red
Green
Blue
Servo
Program
EXEC
TX
Power
RX
Spare
CdS
Mode
OK

48 ロボット動作

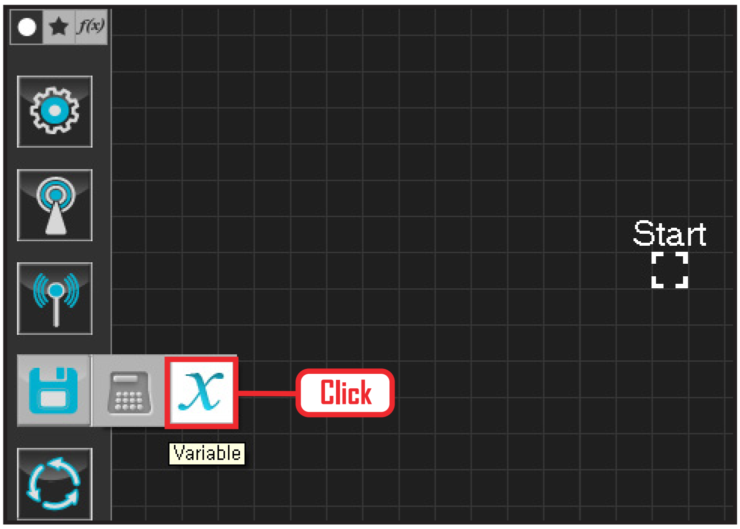
UpボタンはRed
RightボタンはGreen
DownボタンはBlueがオンになります。もう一度押すとオフになります。

Lightの例

例の説明

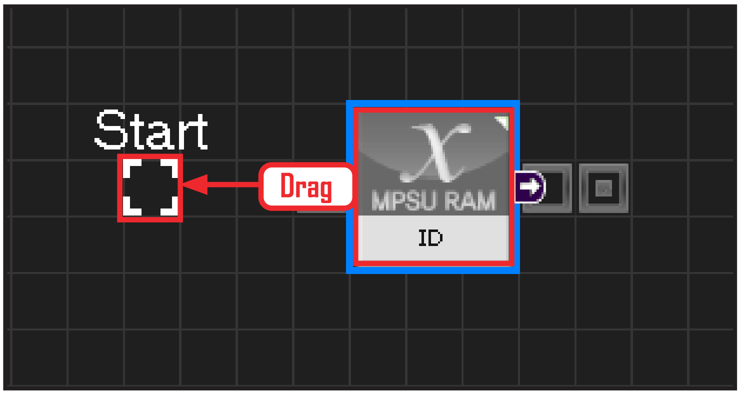
光の強さに応じてロボットのモーターを動作させる例です。

外部の光が暗くなると、ロボットが左腕を上げます。（コントローラー背面のCDSセンサを手のひらで遮ると、光の入ってくる量が減り暗くなるので、ロボットが左腕をあげるようなプログラミングです。）


Start
Click
Variable

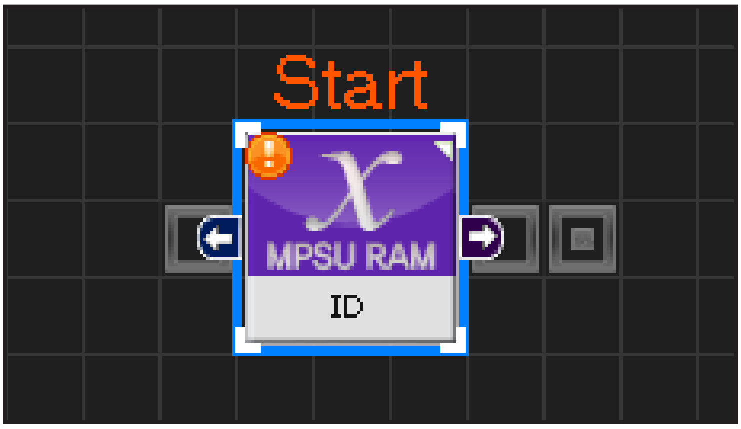
01 変数の指定

Data > Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

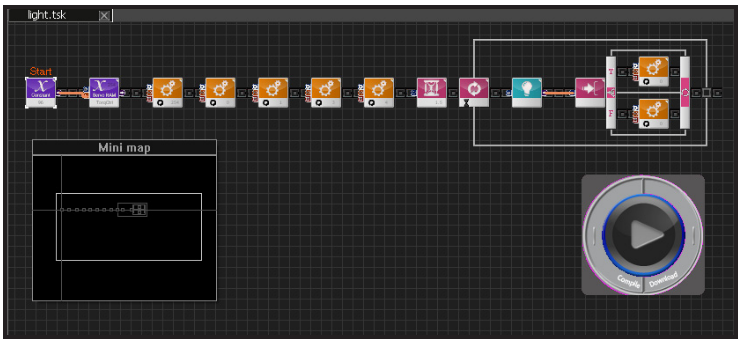
02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。


Start
MPSU RAM
ID

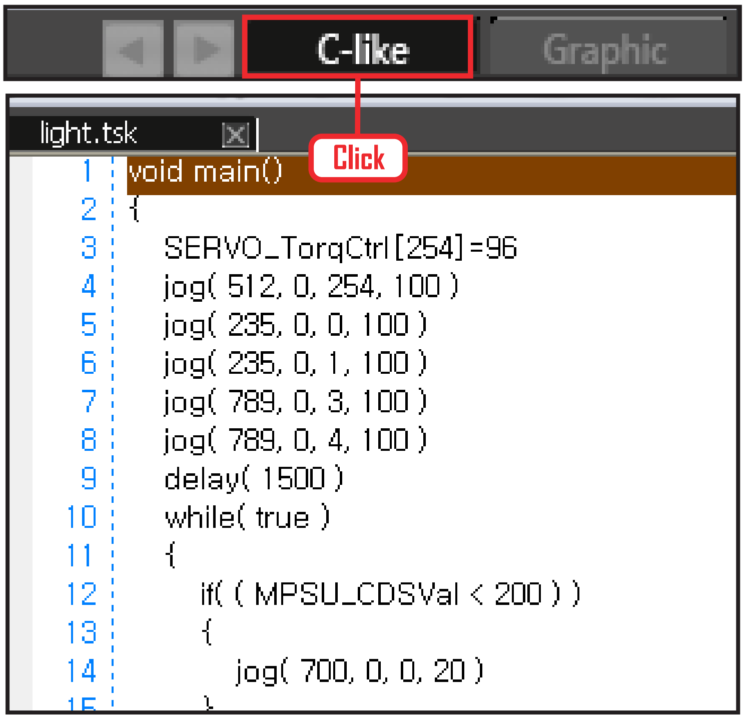
03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。
これはプログラミングが開始されたという意味です。


light.tsk
Start
Mini map

04 完全プログラミング

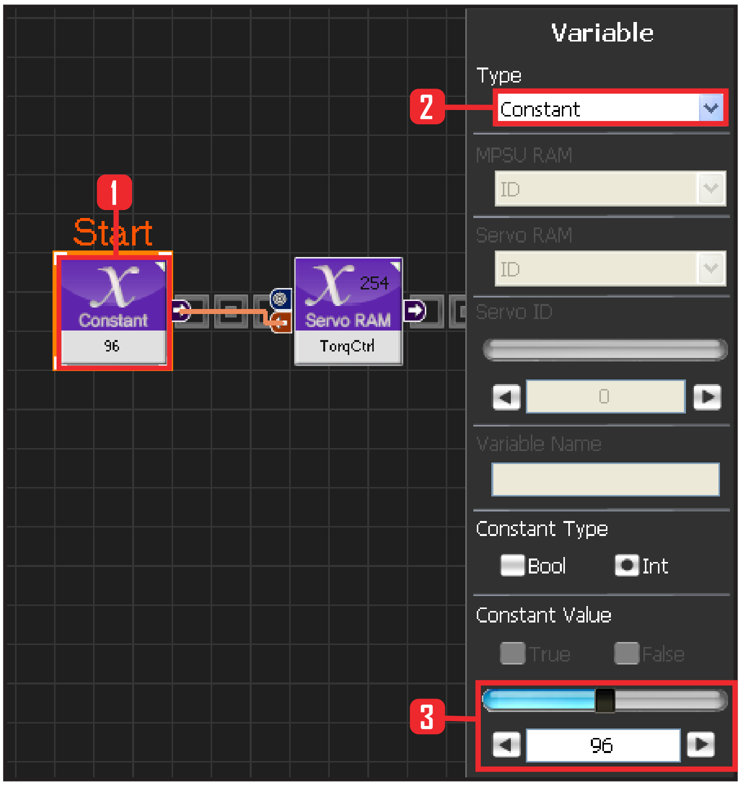
光センサを利用してロボットのモーターを動く全体のプログラムの展開画面です。


C-like
Graphic
light.tsk
Click

```
void main()
{
    SERVO_TorqCtrl[254]=96
    jog( 512, 0, 254, 100 )
    jog( 235, 0, 0, 100 )
    jog( 235, 0, 1, 100 )
    jog( 789, 0, 3, 100 )
    jog( 789, 0, 4, 100 )
    delay( 1500 )
    while( true )
    {
        if( ( MPSU_CDSVal < 200 ) )
        {
            jog( 700, 0, 0, 20 )
        }
```

05 C-Like

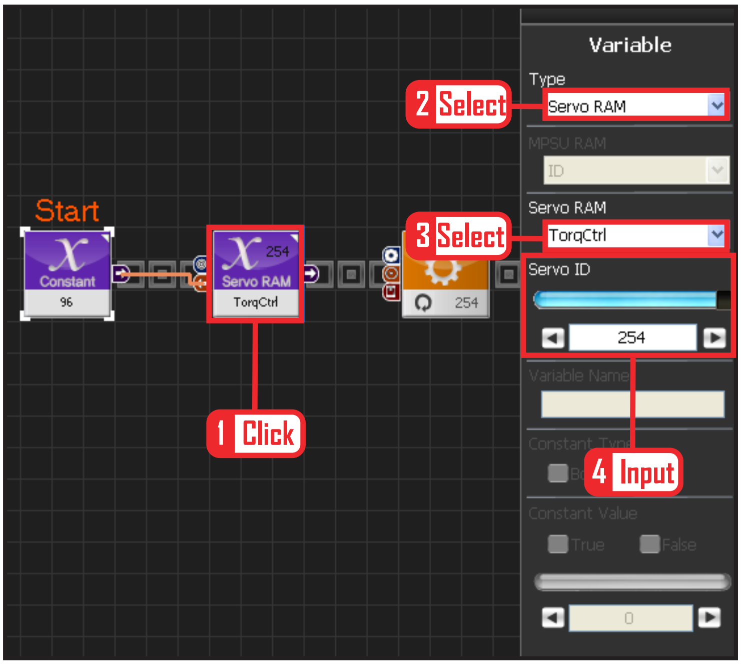
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
光センサを利用して全体プログラムをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。


1
2
3
Start
Constant
96
254
Servo RAM
TorqCtrl
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
True
False

Type: Constant
Constant Type: ○ Bool ● Int
Constant Value: 96

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant　Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。


Start
Constant
96
Servo RAM
254
TorqCtrl
254
1 Click
2 Select
3 Select
4 Input

Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

Constant Type

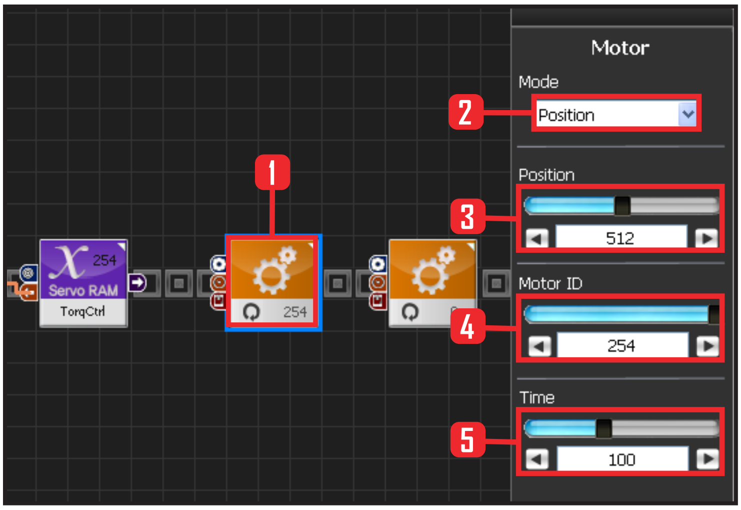
Constant Value: ☐ True ☐ False

0

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type : Servo RAMを選択します。
Servo RAM : TorqCtrlを選択します。
Servo ID : 254を選択します。 254は接続されているすべてのサーボに適用するという意味です。


254
Servo RAM
TorqCtrl
254
1
2
3
4
5

Motor

Mode: Position

Position: 512

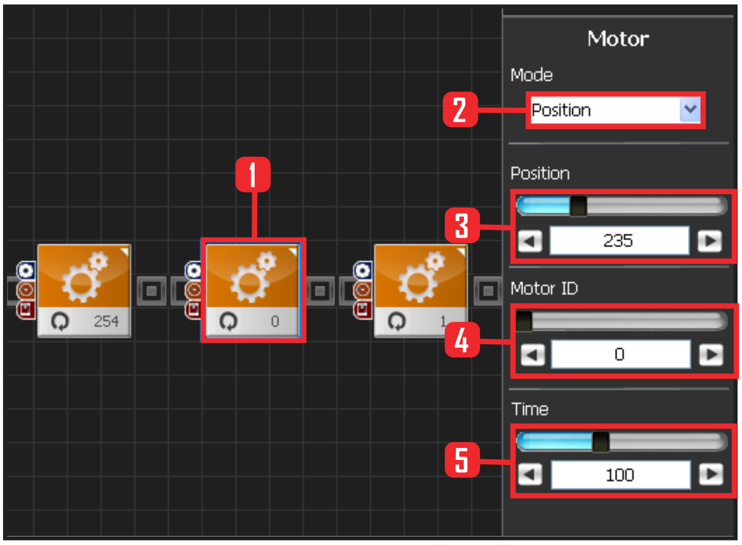
Motor ID: 254

Time: 100

08 すべてのサーボモーターの角度を設定

すべてのサーボモーターの角度を中央に移動させる方法です。

Motion > Motorを選択します。
Mode : Positonを選択します。角度を調整します。
Position : 512に設定します。512番はモーターを中央に移動させるという意味です。
Motor ID : 254に設定します。 254はすべてのモーターに適用するという意味です。
Time : 100に設定します。単位は1あたり11.2msで、100は約1.12秒を意味します。
1.12秒間かけて設定角度に移動させるという意味です。


254
0
1
1
2
3
4
5

Motor

Mode: Position

Position: 235

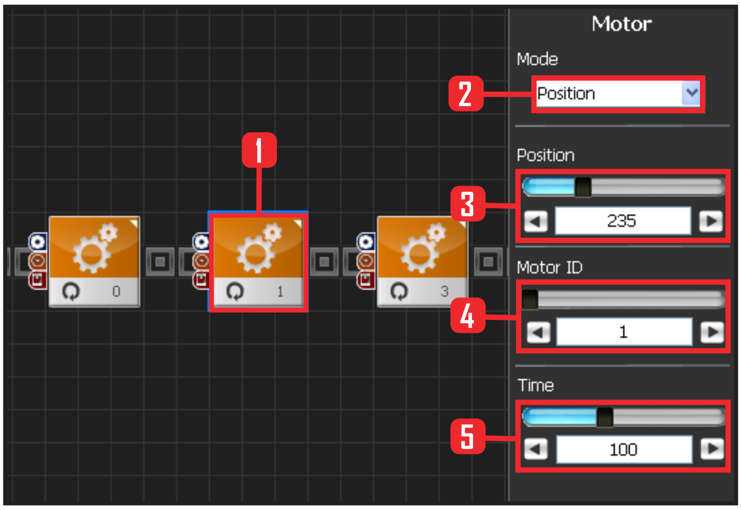
Motor ID: 0

Time: 100

09 モーター0番（右肩）の設定

直立不動の姿勢を作成します。
すべてのロボットのモーターの角度を中央に整列させると、ヒューマノイドは腕を左右に広げます。これを直立不動の姿勢にさせるモーションを容易に作成することができます。

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 235に設定します。 235は水平に広げた右手を垂直に移動させるようにモーターを回すことです。
Motor ID : 0に設定します。右肩のモーターIDが0番です。
Time : 100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

2 Mode: Position

3 Position: 235

4 Motor ID: 1

5 Time: 100

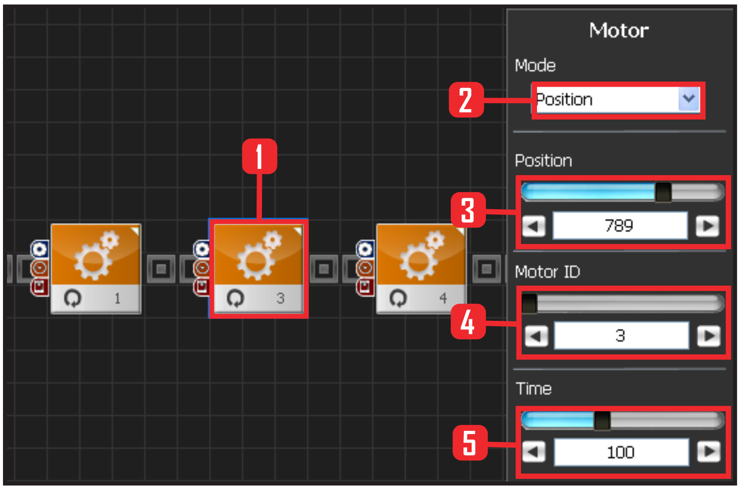
10　モーター1番（右腕の腕）の設定

Mode：Postionを選択します。

Position：235を選択します。 235は水平に広げた腕を垂直に下げる位置です。

Motor ID：1に設定します。右上の腕の肩に接続しているモーターが1番です。

Time：100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

2 Mode: Position

3 Position: 789

4 Motor ID: 3

5 Time: 100

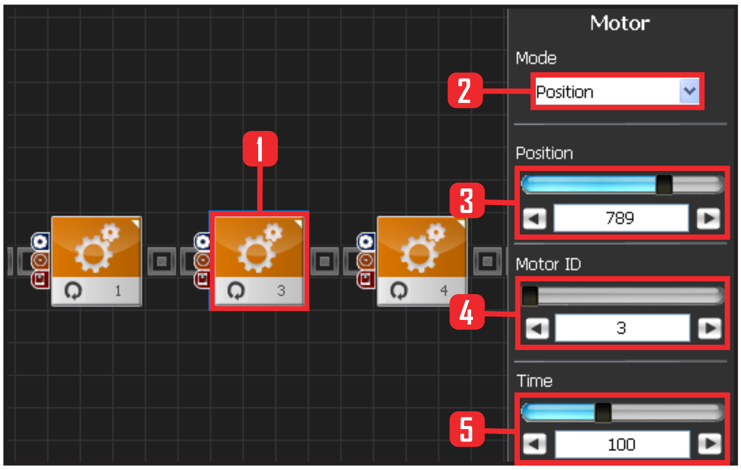
11　モーター3番（左肩）の設定

Mode：Postionを選択します。

Position：789を選択します。 789は水平に広げた左手を垂直に下げる位置です。

Motor　ID：3に設定します。左側の肩のモーターが3番です。

Time：100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

2 Mode: Position

3 Position: 789

4 Motor ID: 3

5 Time: 100

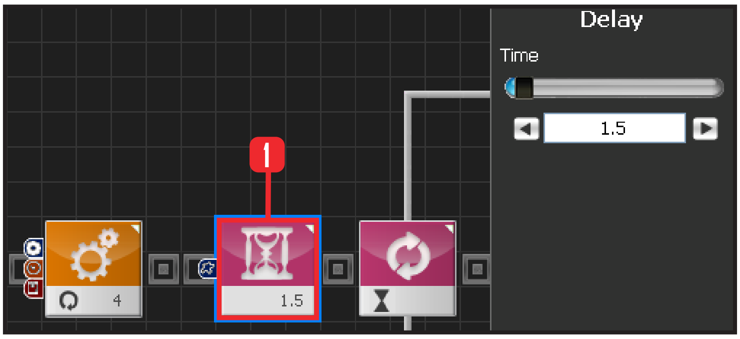
12　モーター4番（左腕の上）の設定

Mode：Postionを選択します。

Position：789を選択します。 789は水平に広げた手を垂直に下げる位置です。

Motor ID：4に設定します。左上の腕の肩に接続しているモーターが4番です。

Time：100に設定します。約1.12秒間かけて設定角度に移動します。

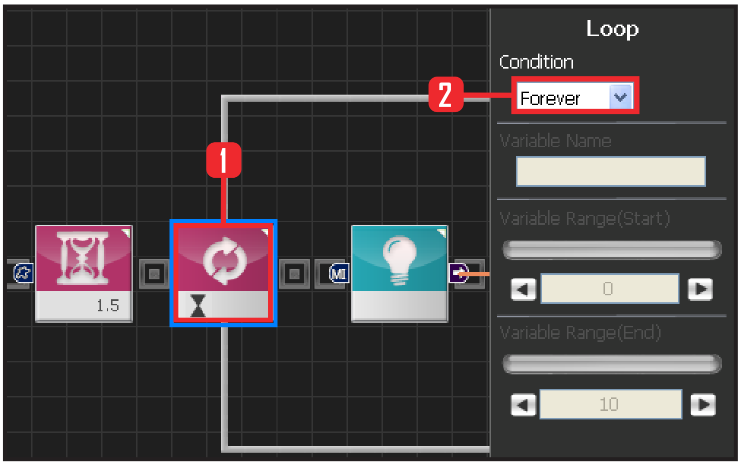


Delay
Time
1.5

13 Delay 遅延させる

ロボットが直立不動の姿勢になるまで待ってから次のモジュールを実行させるために一時的に遅延させる方法です。

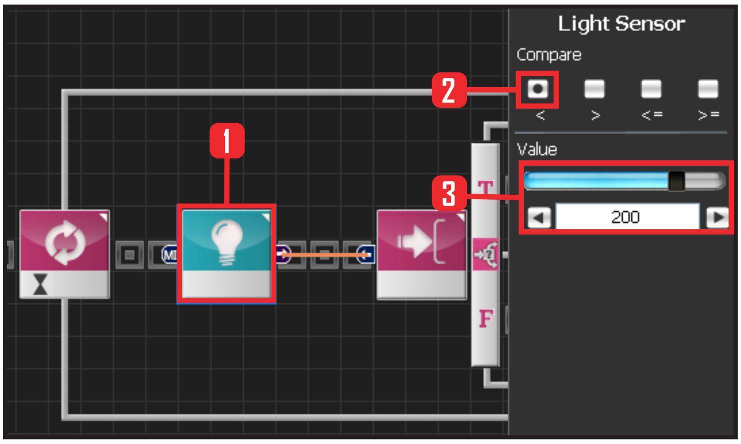
Flow > Delayモジュールを選択します。
Time : 1.5に設定します。単位は秒です。約1.5秒間Delayします。



Loop
Condition
Forever
Variable Name
Variable Range(Start)
0
Variable Range(End)
10

14 Loop 反復文

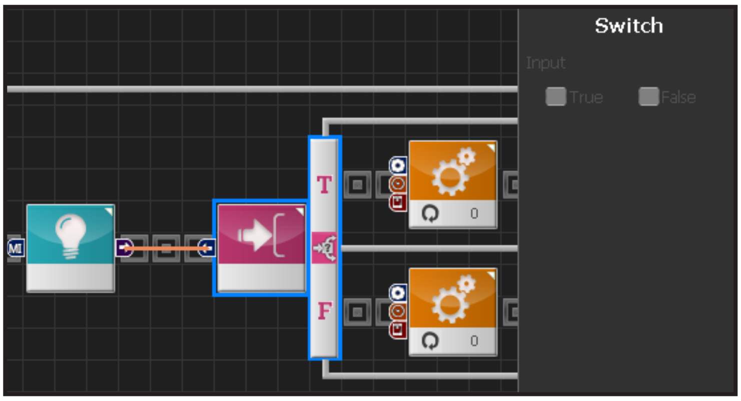
Flow > Loopモジュールを選択します。
ConditionはForeverを選択します。無限ループです。



Light Sensor
Compare
Value
200

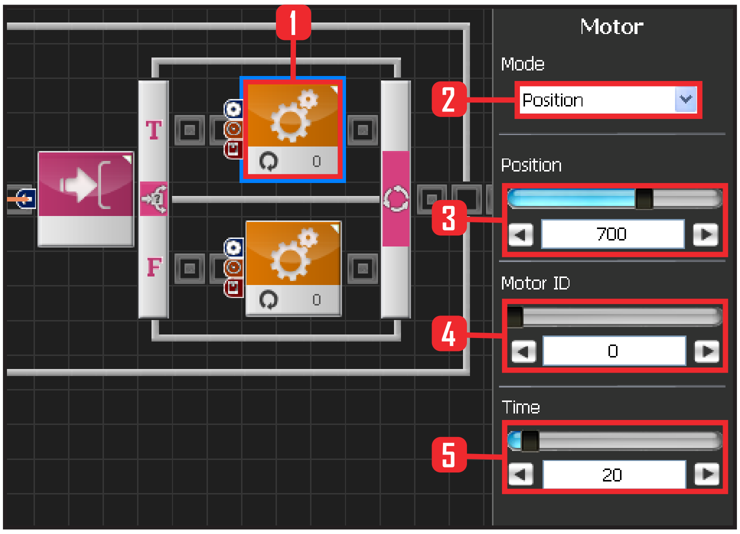
15 Light Sensor 光センサ

Sensor > Lightモジュールを選択します。
Compare : < を選択します。指定した値よりも小さい値の状態を意味します。
Value : 200に設定します。明るさの値が200であることを意味します。
明るさの値が200未満の場合はモジュールのoutputがTrueになります。これ以外の場合はFalseになります。


Switch
Input
☐True ☐False

16 Switch IF 分岐文

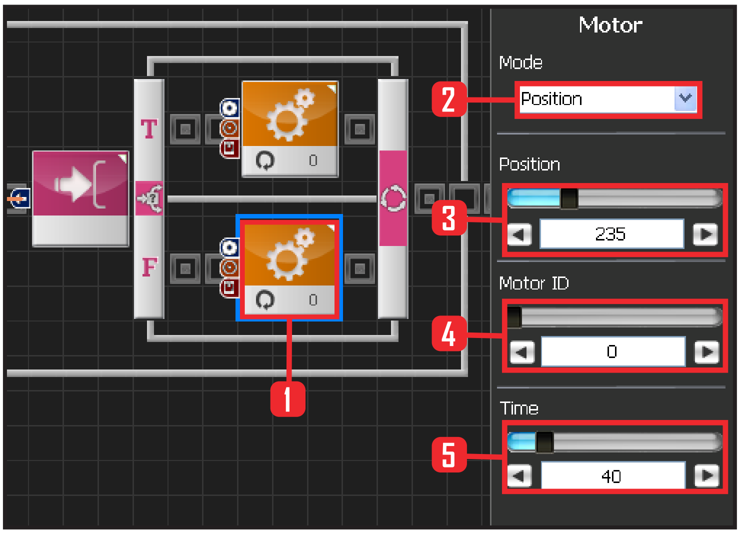
前の値がTrueかFalseの場合、それぞれに該当する結果を実行します。


Motor
Mode
Position
Position
700
Motor ID
0
Time
20

17 モーター0番（右肩）の設定

光の明るさが200より低い場合、つまり、暗い場合（True）は右腕を上げます。明るい場合（False）は右腕を下げます。

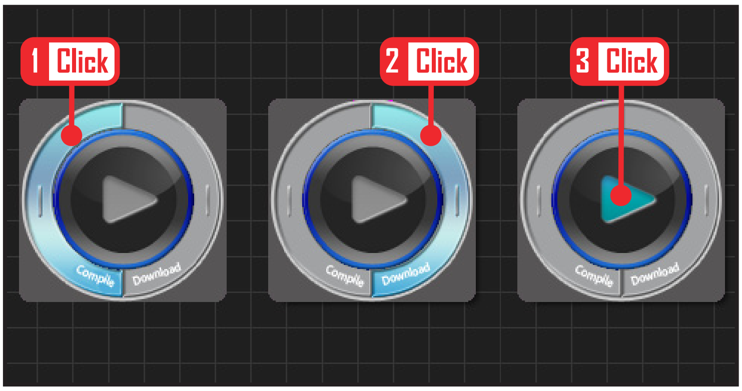
Motion > Moterを選択します。
Mode：Positionを選択します。
Position：700に設定します。 700は腕を前に上げる動作です。
Motor ID：0に設定します。右肩のモータIDが0番です。
Time：20に設定します。


Motor
Mode
Position
Position
235
Motor ID
0
Time
40

18 モーター0番（右肩）の設定

コントローラーのcdsセンサの指を離すと、再び明るくなるので、モーターは直立不動の姿勢に戻ります。

Motion > Moterを選択します。
Mode：Positionを選択します。
Position：235に設定します。235に設定すると、直立不動の姿勢に戻ります。
Motor ID：0に設定します。右肩のモータIDが0番です。
Time：40に設定します。上げた速度よりも少し遅い速度で下げます。


1 Click
2 Click
3 Click
Compile
Download

19 コンパイル、ダウンロード、実行

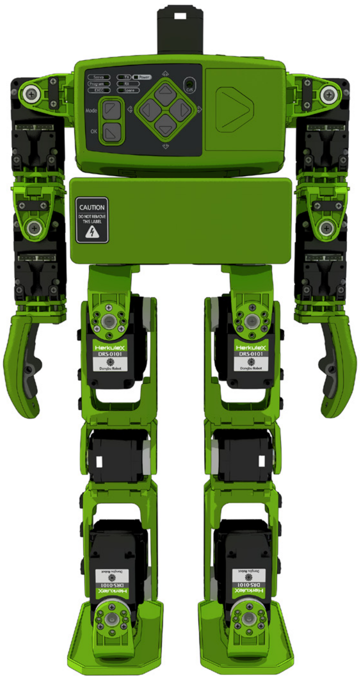
左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。

ロボットは外部の入力待ち状態になります。

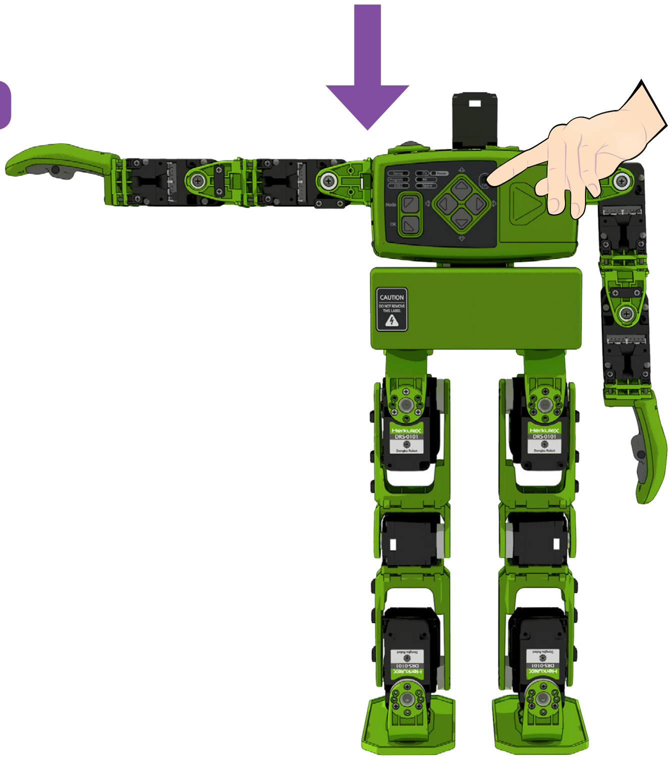
20 ロボット動作

現在は光が明るい状態なので、直立不動の姿勢を維持します。
指でcdsを遮ると、ロボットが右腕を上げます。
指を離すと、ロボットが右腕を下げます。
正確に実行されることを確認します。

1



2

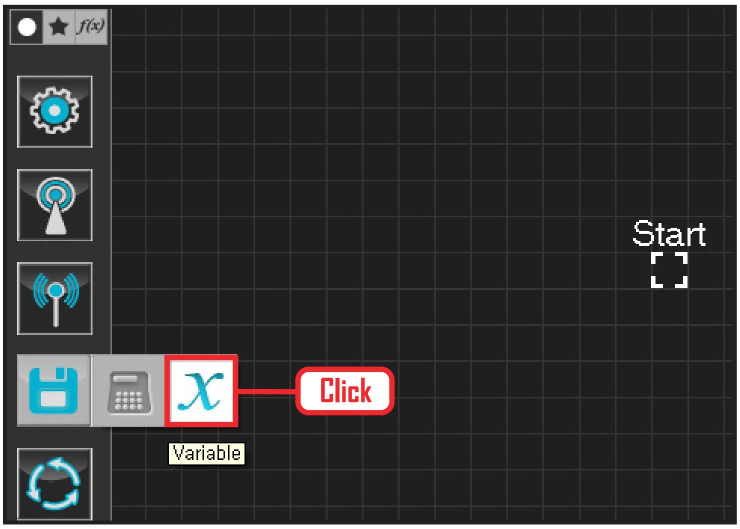


Sound Sensor例

例の説明

Sound SensorはDRCコントローラー内部の両側にあります。

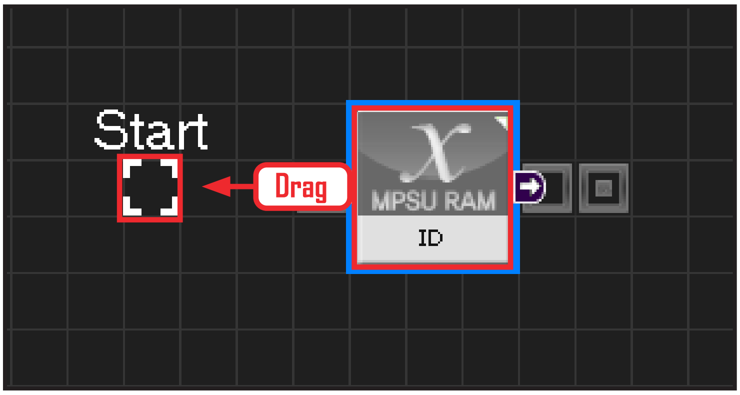
左側から拍手をする場合は左手を上げ、右側から拍手をする場合は右手を上げるプログラミングです。


f(x)
Start
Click
Variable

01 変数の指定

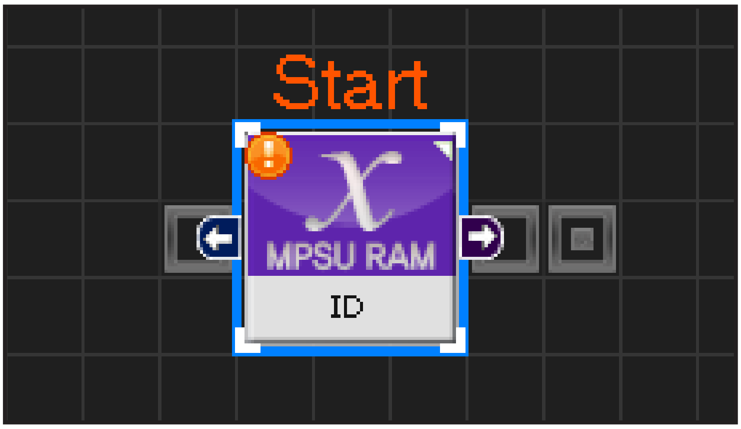
ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data > Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。

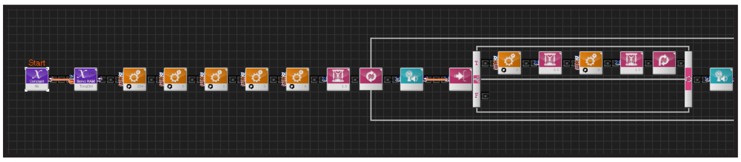

Start
MPSU RAM
ID

03 プログラミングの開始

モジュールとStartPointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。

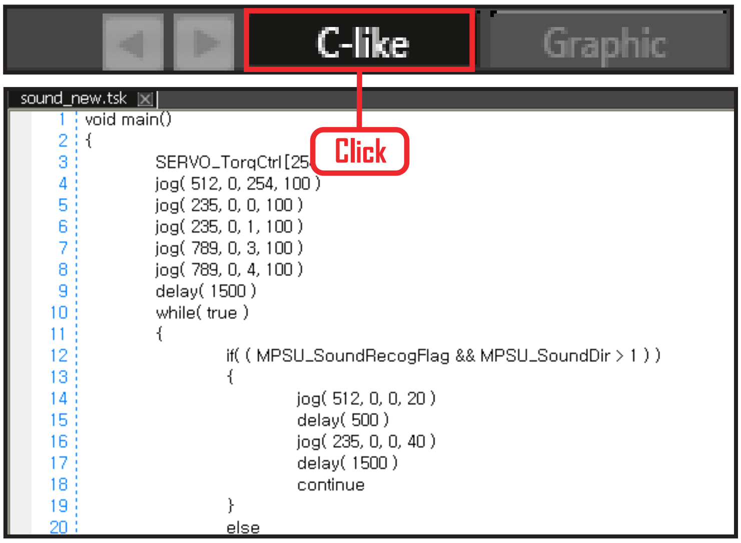
これはプログラミングが開始されたという意味です。

04 完全プログラミング



サウンドセンサを用いてそれぞれのモーターを動作させる全体のプログラミングの画面です。

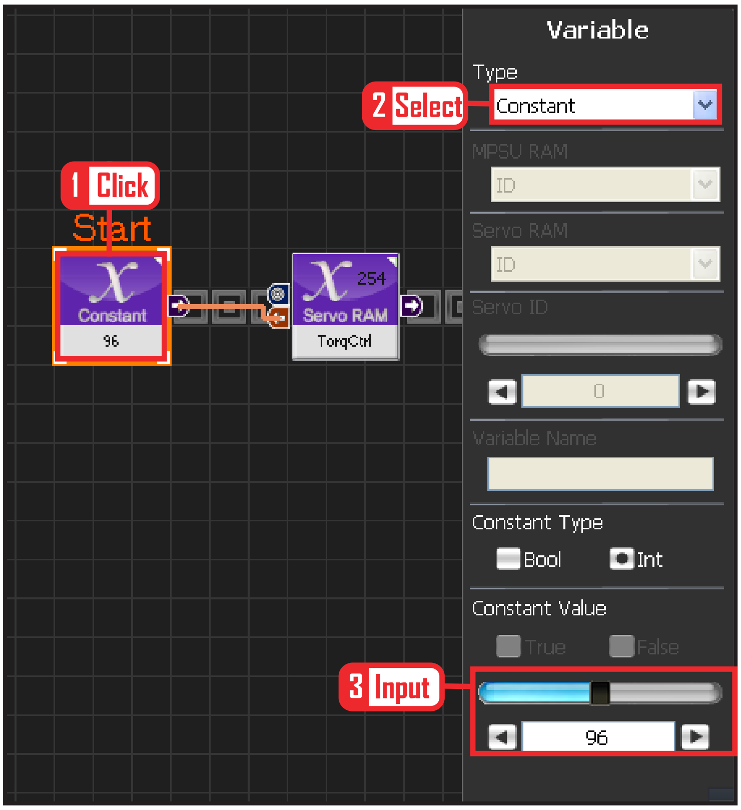
05 C-Like


C-like
Graphic
Click
sound_new.tsk

```
void main()
{
        SERVO_TorqCtrl[25
        jog( 512, 0, 254, 100 )
        jog( 235, 0, 0, 100 )
        jog( 235, 0, 1, 100 )
        jog( 789, 0, 3, 100 )
        jog( 789, 0, 4, 100 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_SoundRecogFlag && MPSU_SoundDir > 1 ) )
                {
                        jog( 512, 0, 0, 20 )
                        delay( 500 )
                        jog( 235, 0, 0, 40 )
                        delay( 1500 )
                        continue
                }
                else
```

右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
サウンドセンサを利用して完全なプログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。

06 定数の設定


1 Click
Start
Constant
96
254
Servo RAM
TorqCtrl
Variable
2 Select
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
3 Input

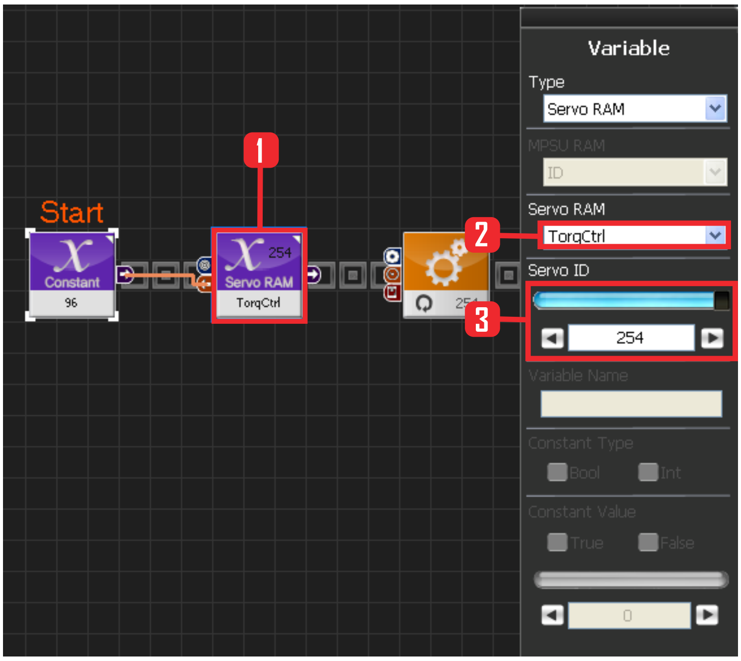
Type: Constant

Constant Type: ☐ Bool ● Int

Constant Value: ☐ True ☐ False

96

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant　Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。

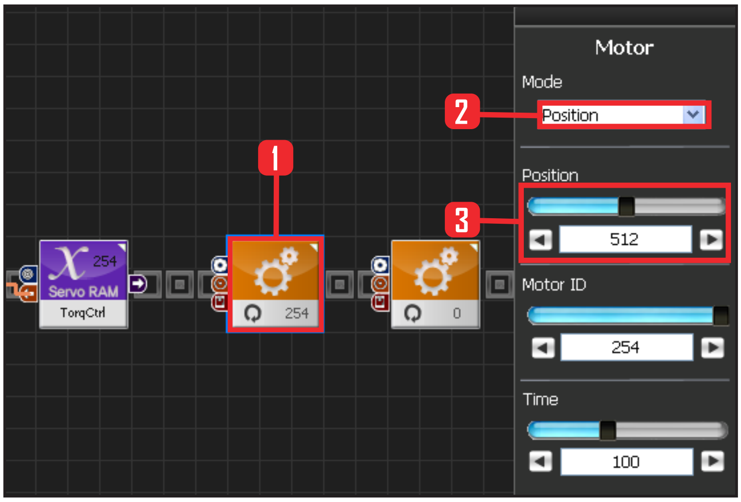


Variable
Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name
Constant Type ☐ Bool ☐ Int
Constant Value ☐ True ☐ False
0

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type : Servo RAMを選択します。
Servo RAM : TorqCtrlを選択します。
Servo ID : 254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

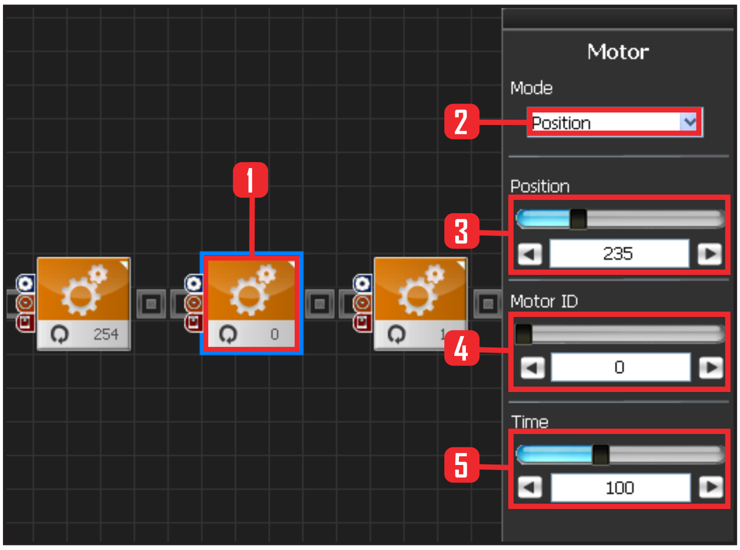


Motor
Mode: Position
Position: 512
Motor ID: 254
Time: 100

08 すべてのサーボモーターの角度を設定

すべてのサーボモーターの角度を中央に移動させる方法です。

Motion > Motorを選択します。
Mode : Positonを選択します。角度を調整します。
Position : 512に設定します。512番はモーターを中央に移動するという意味です。
Motor ID : 254に設定します。 254はすべてのモーターに適用するという意味です。
Time : 100に設定します。単位は1あたり11.2msで、100は約1.12秒を意味します。
1.12秒間かけて設定角度に移動させるという意味です。



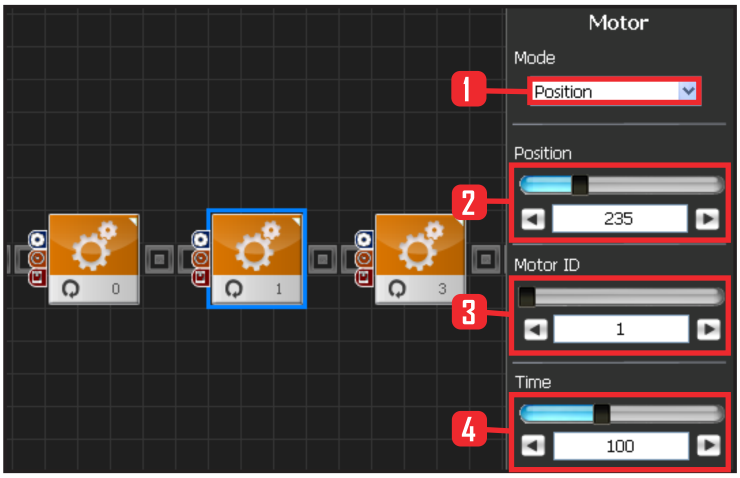
Motor
Mode: Position
Position: 235
Motor ID: 0
Time: 100

09 モーター0番（右肩）の設定

直立不動の姿勢（基本姿勢）を作成する
すべてのロボットのモーターの角度を中央に整列させると、ヒューマノイドは腕を左右に広げます。これを直立不動の姿勢にさせるモーションを容易に作成することができます。

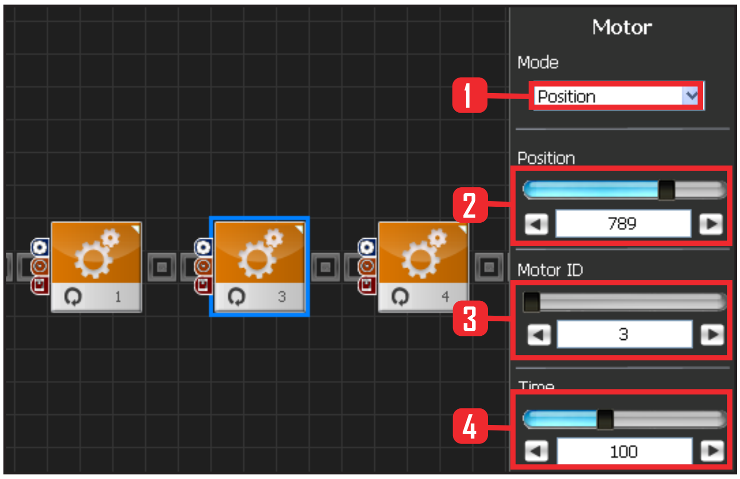
Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 235に設定します。 235は水平に広げた右手を垂直に移動させるようにモーターを回すことです。
Motor ID : 0に設定します。右肩のモーターIDが0番です。
Time : 100に設定します。約1.12秒間かけて設定角度に移動します。

10 モーター1番（右腕）の設定


Motor
Mode: Position
Position: 235
Motor ID: 1
Time: 100

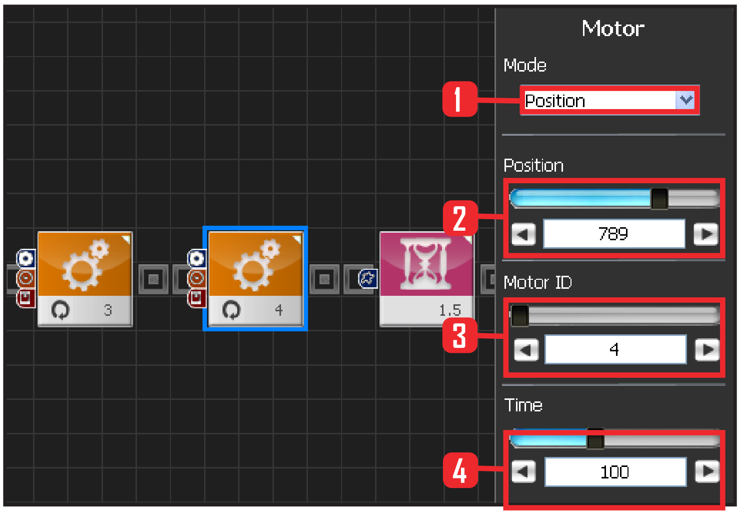
Mode：Postionを選択します。
Position：235を選択します。 235は水平に広げた腕を垂直に下げる位置です。
Motor ID：1に設定します。右上の腕の肩に接続しているモーターが1番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。

11 モーター3番（左肩）の設定

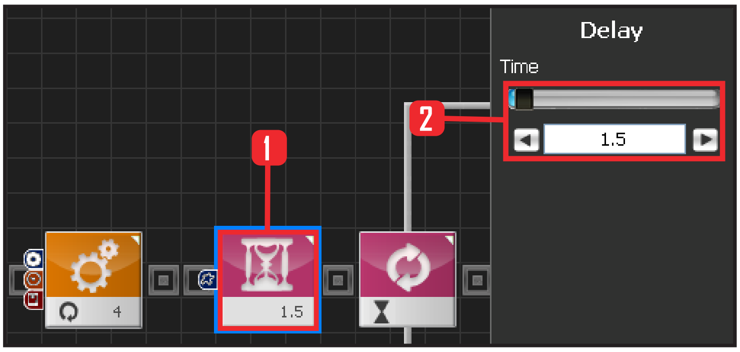

Motor
Mode: Position
Position: 789
Motor ID: 3
Time: 100

Mode：Postionを選択します。
Position：789を選択します。 789は水平に広げた左手を垂直に下げる位置です。
Motor ID：3に設定します。左側の肩のモーターが3番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。

12 モーター4番（左腕）の設定


Motor
Mode: Position
Position: 789
Motor ID: 4
Time: 100

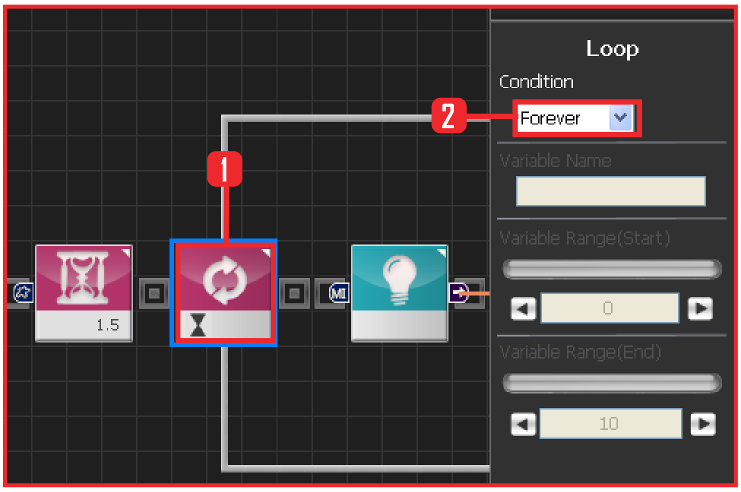
Mode：Postionを選択します。
Position：789を選択します。 789は水平に広げた手を垂直に下げる位置です。
Motor ID：4に設定します。左上の腕の肩に接続しているモーターが4番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。


Delay
Time
1
2
4
1.5
1.5

13 Delay 遅延させる

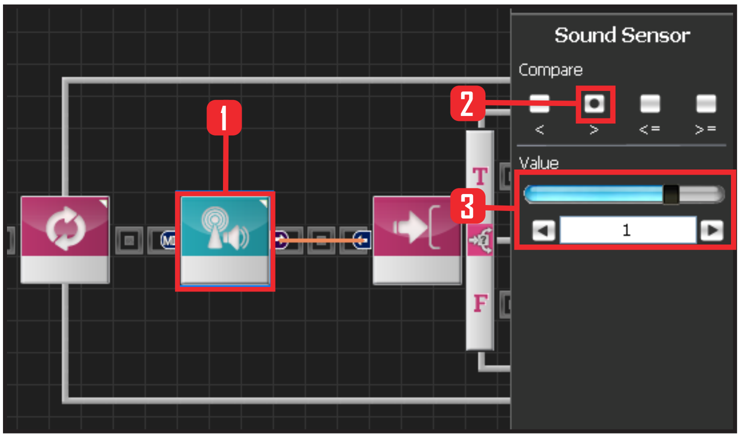
ロボットが直立不動の姿勢になるまで待ってから次のモジュールを実行させるために一時的に遅延させる方法です。

Flow > Delayモジュールを選択します。
Time : 1.5に設定します。単位は秒です。約1.5秒間Delayします。


Loop
Condition
Forever
Variable Name
Variable Range(Start)
0
Variable Range(End)
10
1
2
1.5

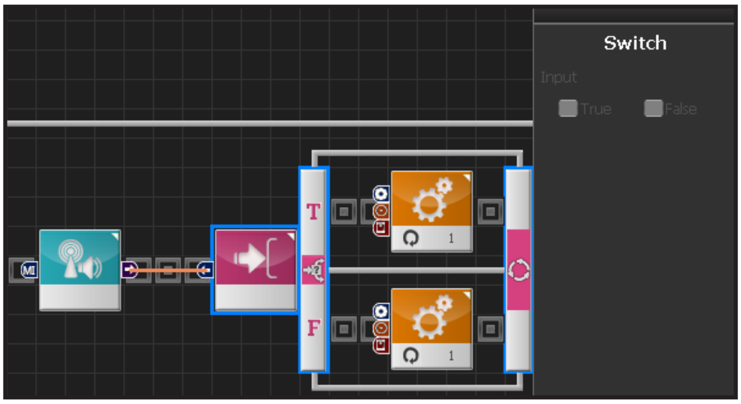
14 Loop 反復文

Flow > Loopモジュールを選択します。
ConditionでForeverを選択します。無限ループです。


Sound Sensor
Compare
<
>
<=
>=
Value
1
1
2
3
T
F

15 Sound Sensor

Sensor > Sound Sensorモジュールを選択します。
Compare : >を選択します。指定した値より大きい時を意味します。
Value : 0に設定します。音の方向は-2から2までであり、負の数は左から音が発生したことを、正の数は右から音が発生したことを意味します。
0より大きいということは右から音が入ってきたことを意味します。音の方向が右の場合はOutputがTrueになり、その他の場合はFalseになります。


Switch

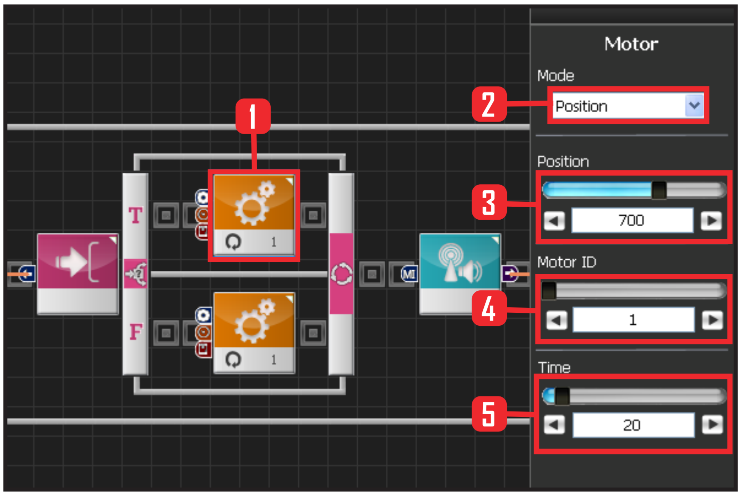
Input

☐ True ☐ False

16 Switch IF 分岐文

前の値がTrueかFalseの場合、それぞれに該当する結果を実行します。

右音の場合はTrue、それ以外の場合はFalse。


Motor

Mode: Position

Position: 700

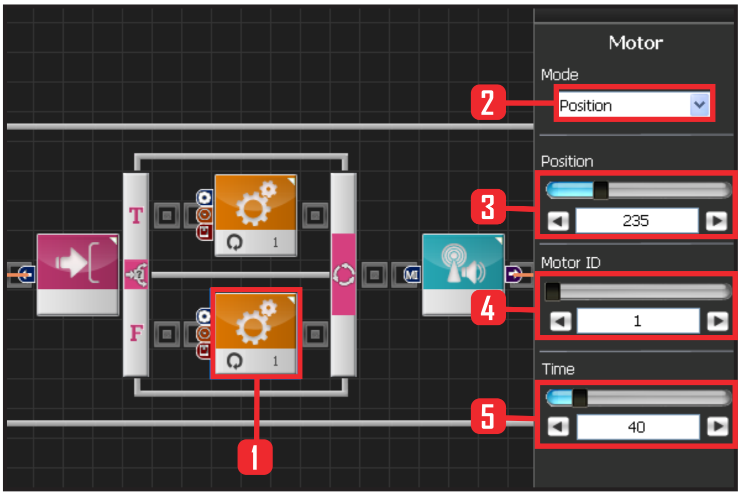
Motor ID: 1

Time: 20

17 モーター1番(腕)の設定

右音の場合(True)、腕を上げます。

Motion > Motorを選択します。
Mode:Positionを選択します。
Position:700に設定します。 700は腕を横に上げる動作です。
Motor ID:1に設定します。右腕のモーターIDが1番です。
Time:20に設定します。


Motor

Mode: Position

Position: 235

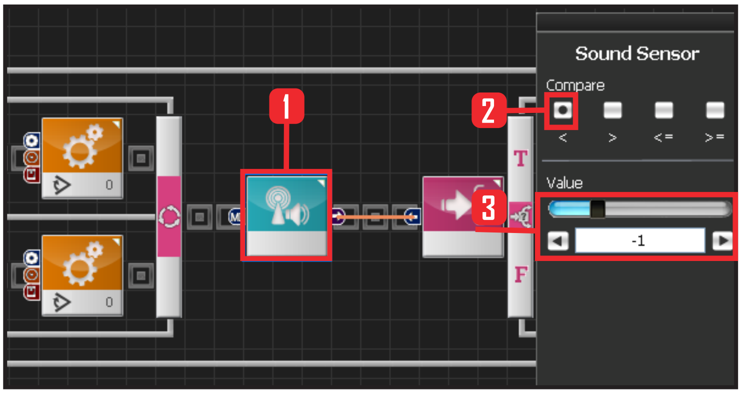
Motor ID: 1

Time: 40

18 モーター1番(腕)の設定

音がない場合または右以外の方向から出た(False)場合は現在の腕を下げた状態を維持します。

Motion > Motorを選択します。
Mode:Positionを選択します。
Position:235に設定します。 235は直立不動の姿勢を維持します。腕が上がっている場合は直立不動の姿勢に戻ります。
Motor ID:1に設定します。右腕のモーターIDが1番です。
Time:40に設定します。



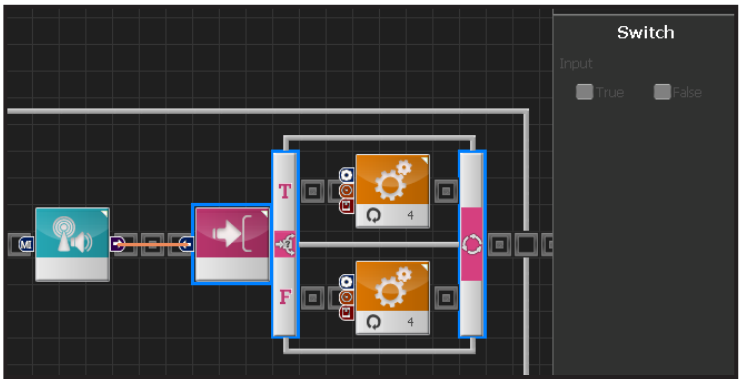
Sound Sensor

Compare
○ < ☐ > ☐ <= ☐ >=

Value
-1

19 Sound Sensor

Sensor > Sound Sensorモジュールを選択します。
Compare: < を選択します。指定した値より小さい時を意味します。
Value:0に設定します。音の方向は-2から2までであり、負の数は左から音が発生したことを、正の数は右から音が発生したことを意味します。
0より大きいということは右から音が入ってきたことを意味します。音の方向が右の場合はOutputがTrueになり、その他の場合はFalseになります。



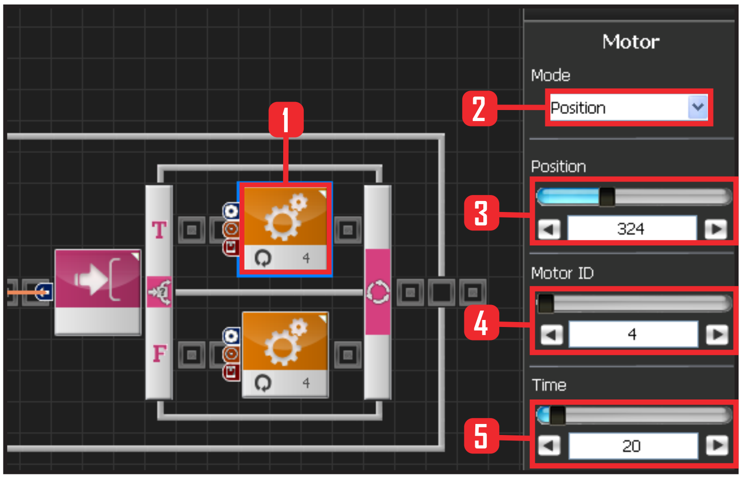
Switch

Input
☐ True ☐ False

20 Switch IF 分岐文

前の値がTrueかFalseの場合、それぞれに該当する結果を実行します。

左音の場合はTrue、それ以外の場合はFalse。



Motor

Mode
Position

Position
324

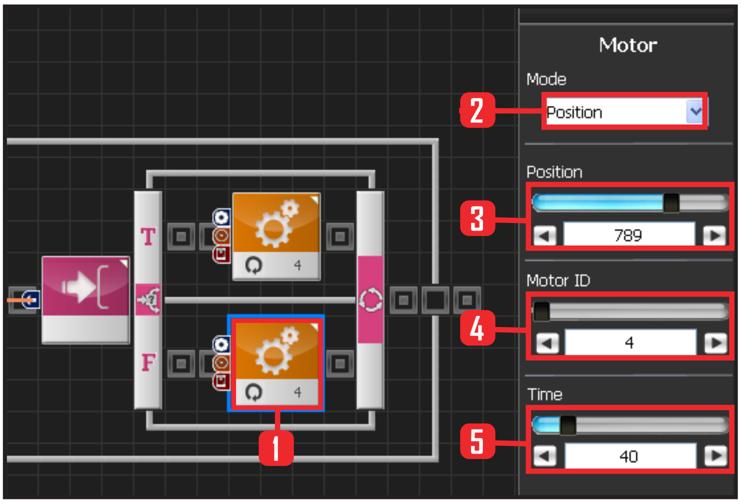
Motor ID
4

Time
20

21 モーター4番（腕）の設定

左音の場合（True）、腕を上げます。

Motion > Motorを選択します。
Mode:Positionを選択します。
Position:324に設定します。　324は腕を横に上げる動作です。
Motor　ID:4に設定します。左腕のモーターIDが4番です。
Time:20に設定します。

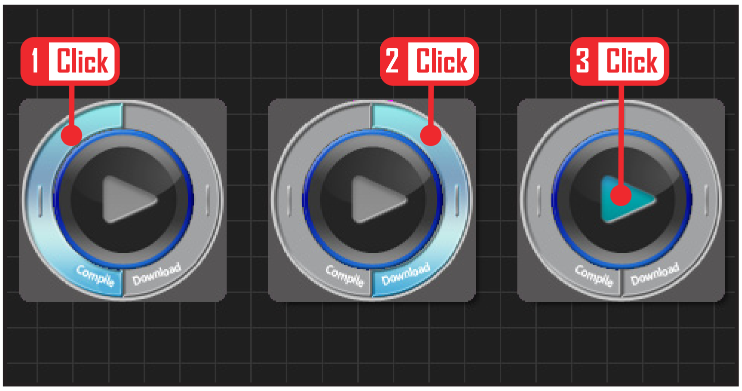


Motor
Mode: Position
Position: 789
Motor ID: 4
Time: 40

22 モーター4番（腕）の設定

音がない場合または左以外の方向から出た(False)場合は現在の腕を下げた状態を維持します。

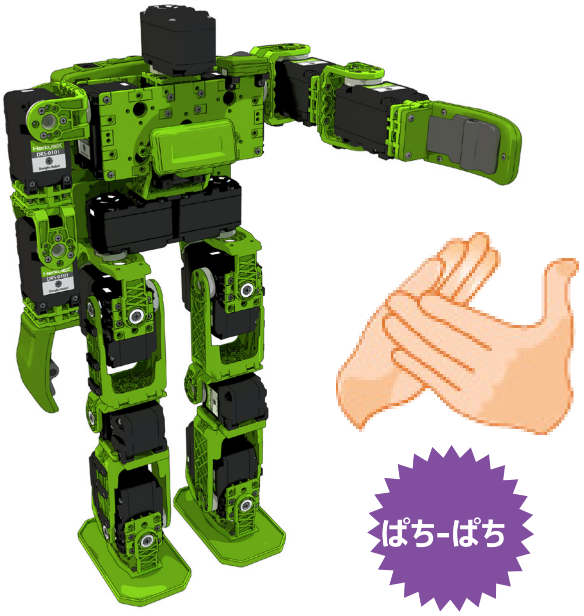
Motion > Motorを選択します。
Mode:Positionを選択します。
Position:789に設定します。　789は直立不動の姿勢を維持します。腕が上がっている場合は直立不動の姿勢に戻ります。
MotorID:4に設定します。右腕のモーターIDが4番です。
Time:40に設定します。


1 Click
2 Click
3 Click
Compile
Download

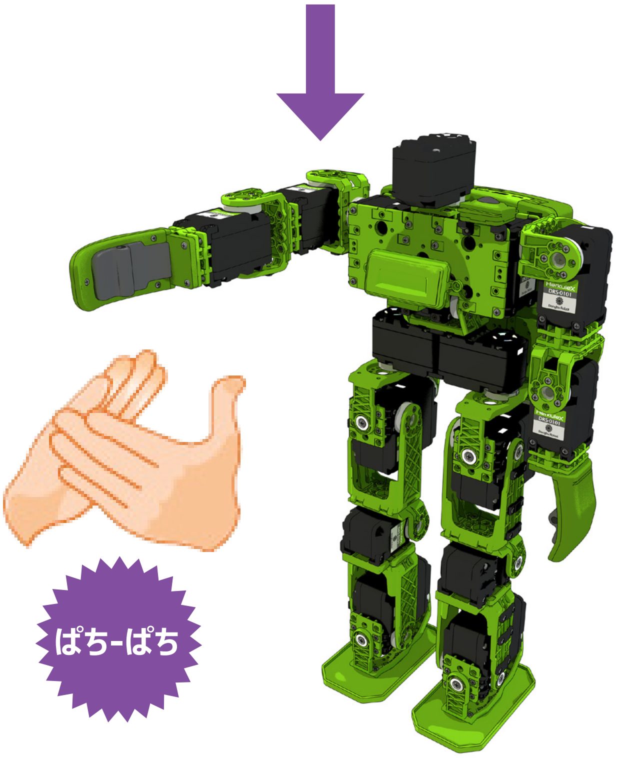
23 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。

1



2



24 ロボット動作

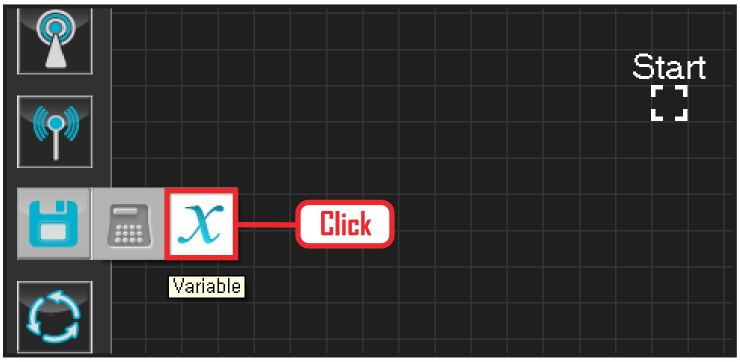
ロボットの右側から拍手する場合は右腕を上げて、ロボットの左側から拍手する場合は左腕を上げます。

Sound Sensor（深化）の例

例の説明

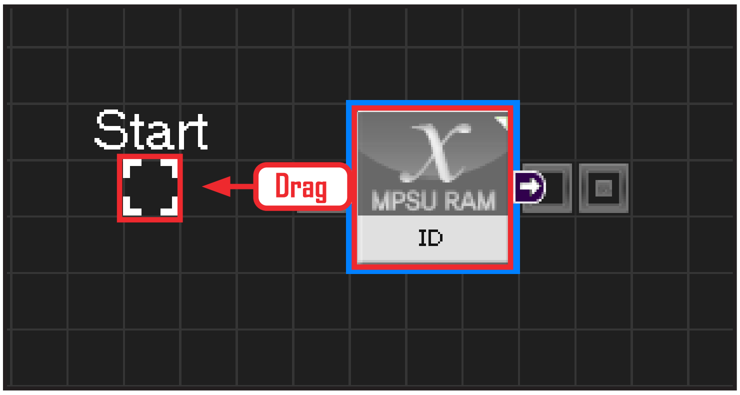
Sound　SensorはDRCコントローラー内部の両側にあります。左側で拍手をすると、左手を上げます。右側で拍手をすると、右手を上げるプログラミングです。（Sound Sensor Sample1番）

周囲の騒音が多い時は両側からの音をそれぞれ区別することが難しくなります。片側で拍手をしても両手を上げたり、不規則な反応をします。周囲の騒音が多い時も、ロボットが正確な反応をするために、より細分化されたプログラミングをする必要があります。最初に音が入力された時に他の音が入力されないように、強制的にDelayを与えることによって、一度拍手をした時に一度だけ腕を上げるように精度を向上させるプログラムです。（Sound Sensor Sample 2番）



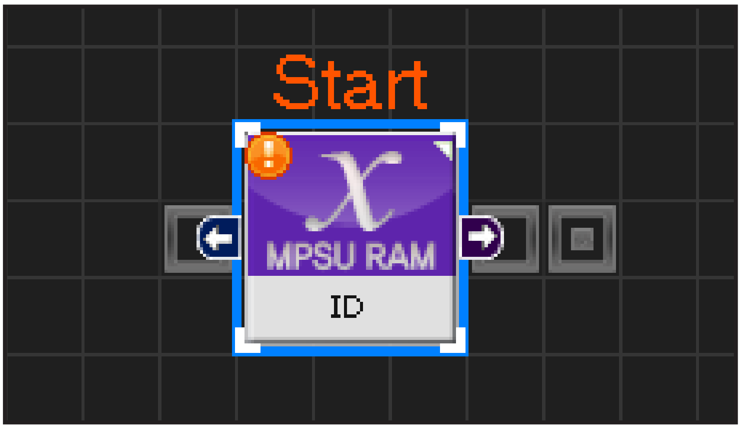
01 変数の指定

ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。
Data > Variableモジュールをクリックします。



02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせせます。

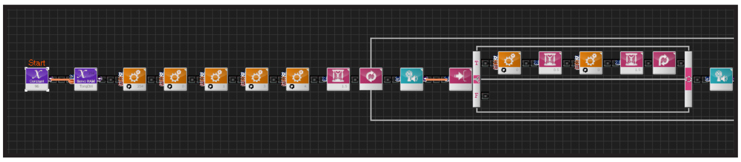


03 プログラミングの開始

モジュールとStartPointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。
これはプログラミングが開始されたという意味です。

04 完全プログラミング

サウンドセンサの感度を高め、ロボットがより正確に反応するようにプログラミングした全体のプログラミングの画面です。



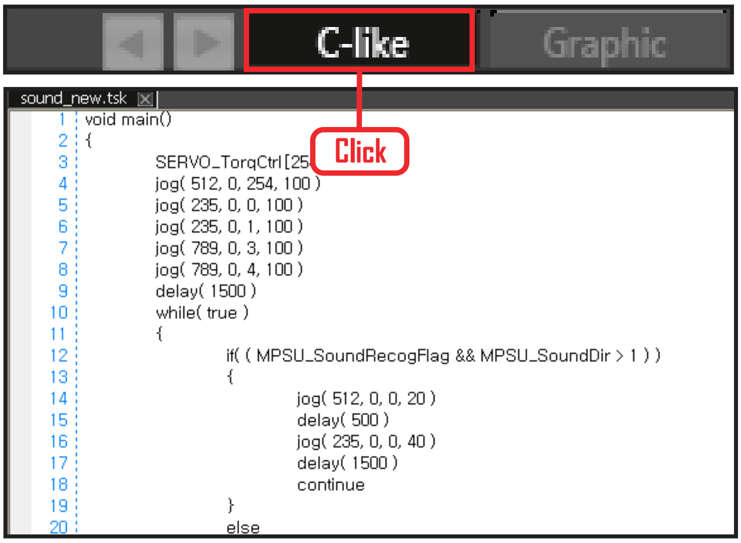
05 C-Like

右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。

サウンドセンサを利用して完全なプログラミングをする画面です。

Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。

各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。


C-like
Graphic
Click
sound_new.tsk

```
void main()
{
        SERVO_TorqCtrl[25
        jog( 512, 0, 254, 100 )
        jog( 235, 0, 0, 100 )
        jog( 235, 0, 1, 100 )
        jog( 789, 0, 3, 100 )
        jog( 789, 0, 4, 100 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_SoundRecogFlag && MPSU_SoundDir > 1 ) )
                {
                        jog( 512, 0, 0, 20 )
                        delay( 500 )
                        jog( 235, 0, 0, 40 )
                        delay( 1500 )
                        continue
                }
                else
```

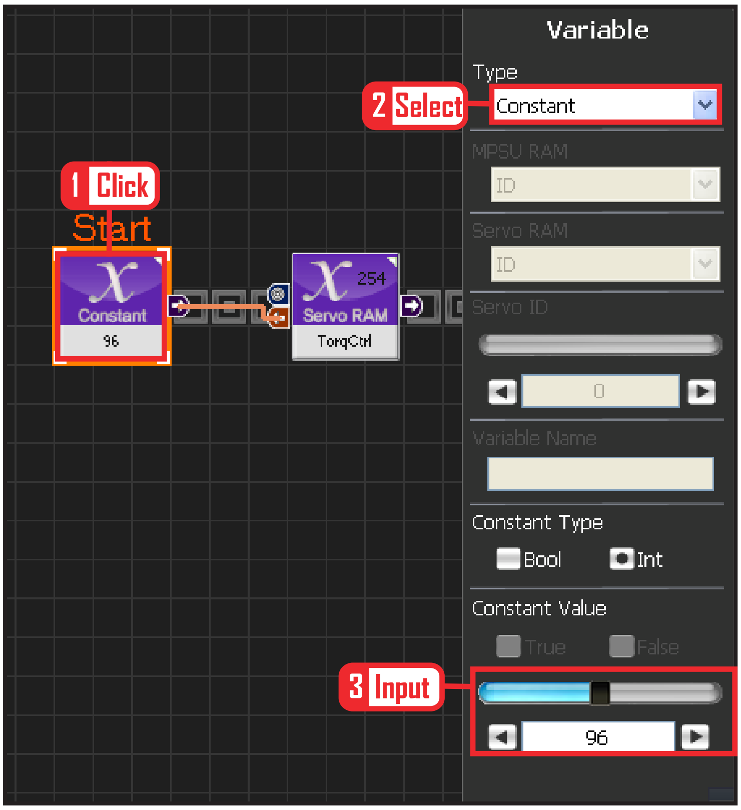
06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。

Variable TypeでConstantを選択します。

プロパティー中のConstant　Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かすことができる状態になります。

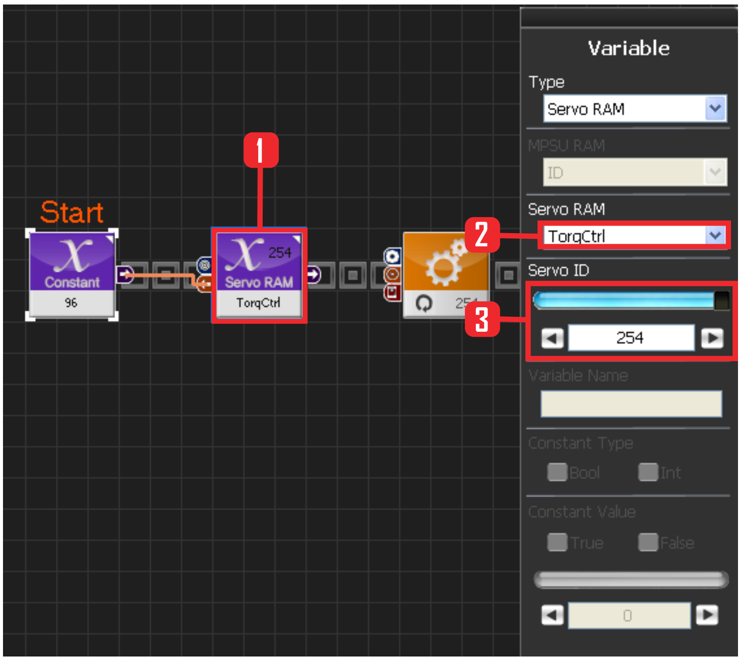
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。


1 Click
Start
Constant
96
254
Servo RAM
TorqCtrl
2 Select
3 Input
Variable
MPSU RAM
Servo RAM
Servo ID
Variable Name
True
False

Type: Constant

Constant Type: ☐ Bool ● Int

Constant Value: 96



Variable

Type: Servo RAM

MPSU RAM: ID

Servo RAM: TorqCtrl

Servo ID: 254

Variable Name

Constant Type: ☐ Bool ☐ Int

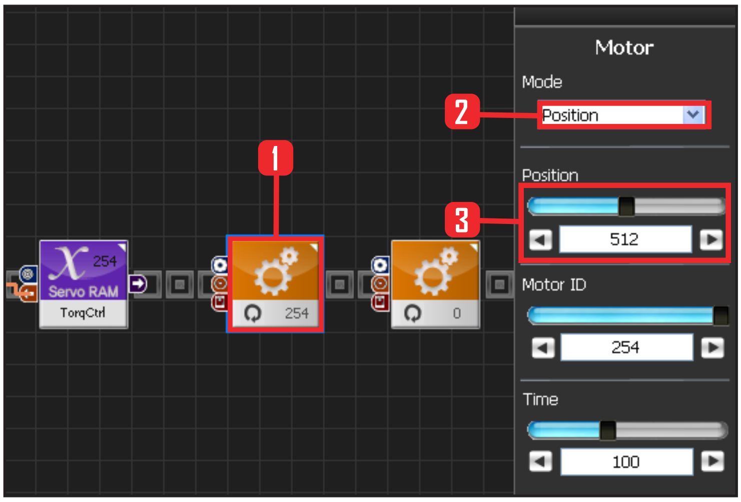
Constant Value: ☐ True ☐ False

0

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type : Servo RAMを選択します。
Servo RAM : TorqCtrlを選択します。
Servo ID : 254を選択します。 254は接続されているすべてのサーボに適用するという意味です。



Motor

Mode: Position

Position: 512

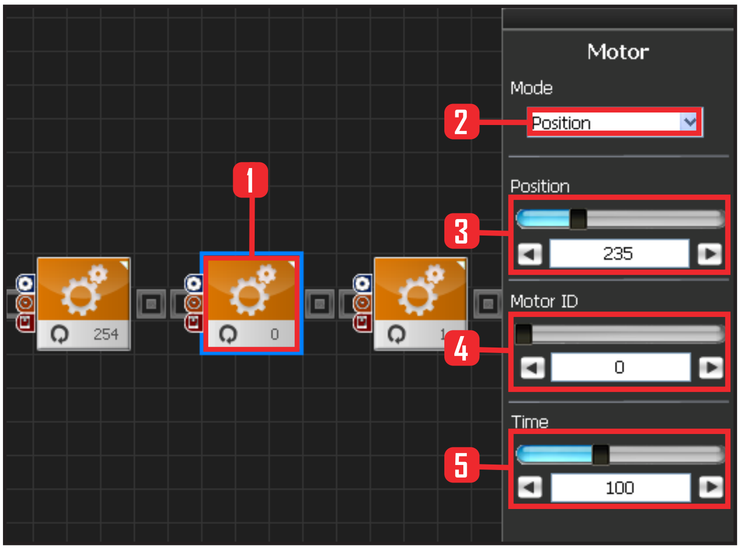
Motor ID: 254

Time: 100

08 すべてのサーボモーターの角度を設定

すべてのサーボモーターの角度を中央に移動する方法です。

Motion > Motorを選択します。
Mode : Positonを選択します。角度を調整します。
Position : 512に設定します。512番はモーターを中央に移動するという意味です。
Motor ID : 254に設定します。 254はすべてのモーターに適用するという意味です。
Time : 100に設定します。単位は1あたり11.2msで、100は約1.12秒を意味します。
1.12秒間かけて設定角度に移動させるという意味です。



Motor

Mode: Position

Position: 235

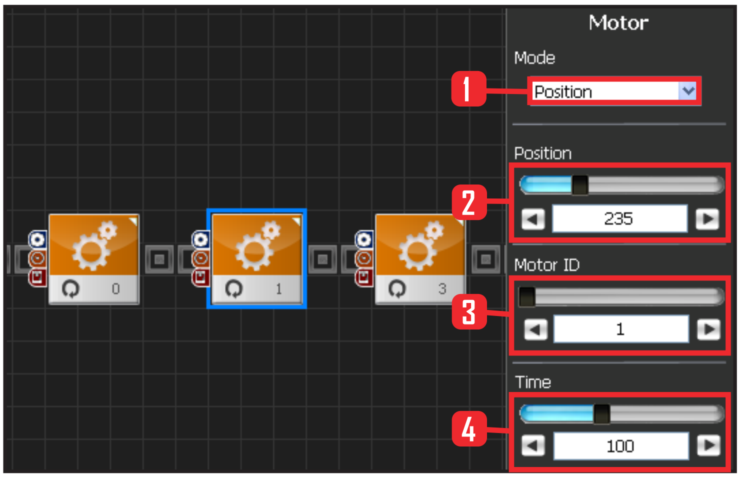
Motor ID: 0

Time: 100

09 モーター0番（右肩）の設定

直立不動の姿勢（基本姿勢）を作成します。
すべてのロボットのモーターの角度を中央に整列させると、ヒューマノイドは腕を左右に広げます。これを直立不動の姿勢にさせるモーションを容易に作成することができます。

Motion > Motorを選択します。
Mode : Positionを選択します。
Position : 235に設定します。 235は水平に広げた右手を垂直に移動させるようにモーターを回すことです。
Motor ID : 0に設定します。右肩のモーターIDが0番です。
Time : 100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

Mode: Position

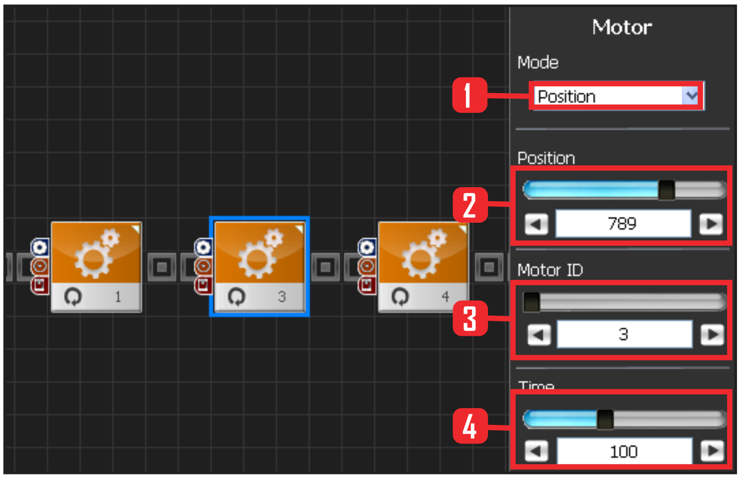
Position: 235

Motor ID: 1

Time: 100

10 モーター1番（右腕）の設定

Mode：Postionを選択します。
Position：235に設定します。 235は水平に広げた腕を垂直に下げる位置です。
Motor ID：1に設定します。右上の腕の肩に接続しているモーターが1番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

Mode: Position

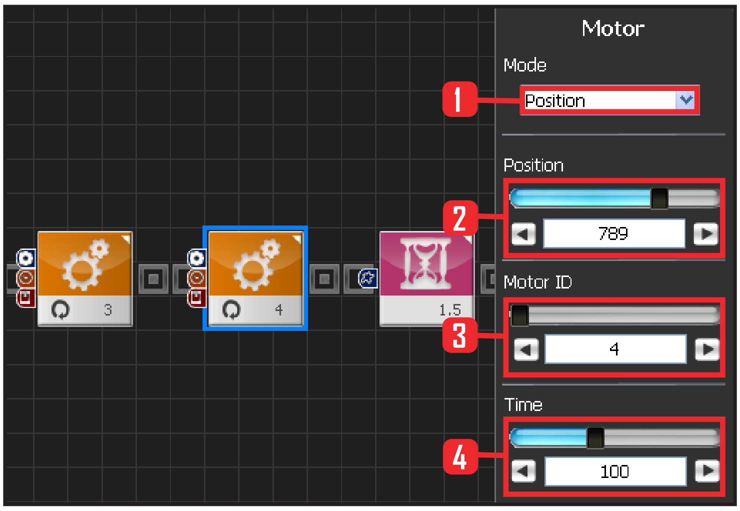
Position: 789

Motor ID: 3

Time: 100

11 モーター3番（左肩）の設定

Mode：Postionを選択します。
Position：789に設定します。 789は水平に広げた左手を垂直に下げる位置です。
Motor ID：3に設定します。左側の肩のモーターが3番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。


Motor

Mode: Position

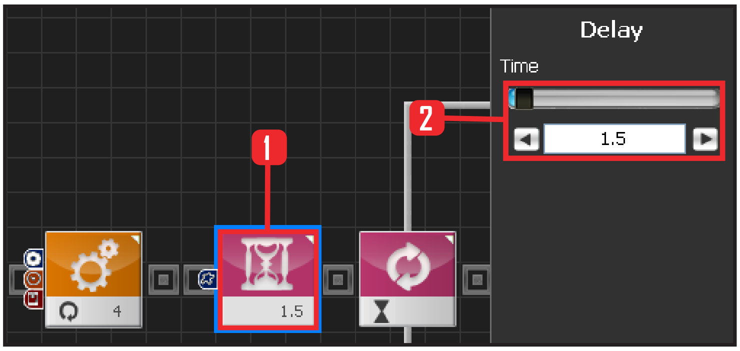
Position: 789

Motor ID: 4

Time: 100

12 モーター4番（左腕）の設定

Mode：Postionを選択します。
Position：789に設定します。 789は水平に広げた手を垂直に下げる位置です。
Motor ID：4に設定します。左上の腕の肩に接続しているモーターが4番です。
Time：100に設定します。約1.12秒間かけて設定角度に移動します。

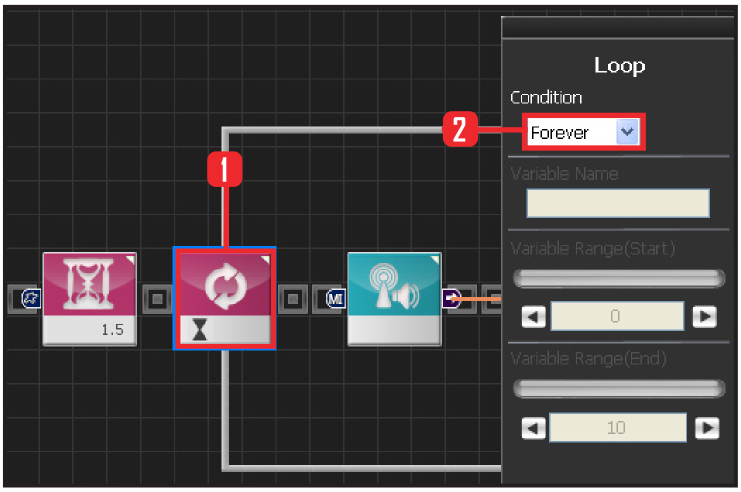


Delay
Time
1.5

13 Delay 遅延させる

ロボットが直立不動の姿勢になるまで待ってから次のモジュールを実行させるために一時的に遅延させる方法です。

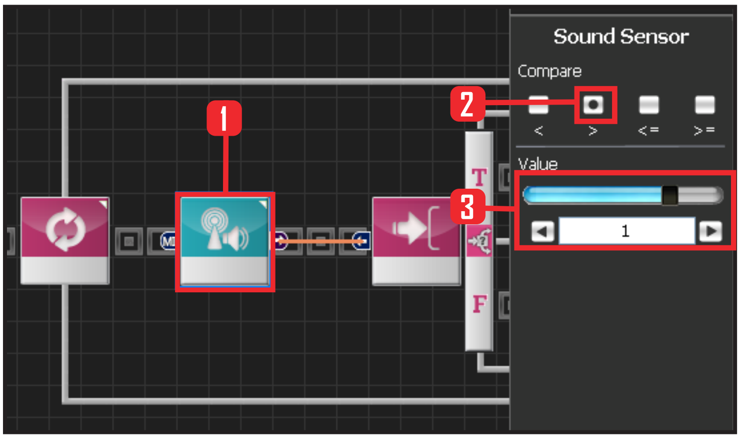
Flow > Delayモジュールを選択します。
Time : 1.5に設定します。単位は秒です。約1.5秒間Delayします。



Loop
Condition
Forever
Variable Name
Variable Range(Start)
0
Variable Range(End)
10

14 Loop 反復文

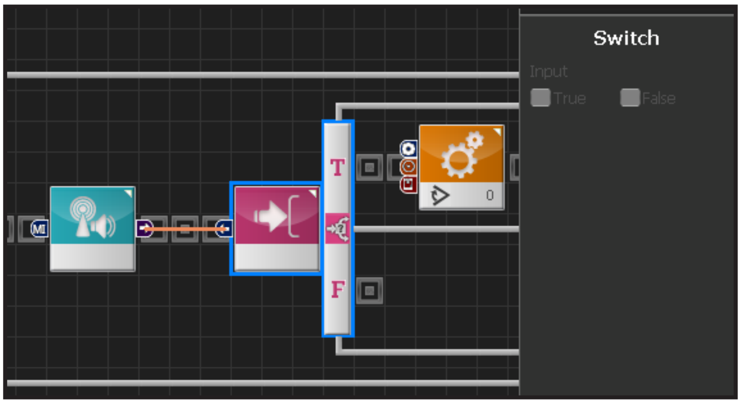
Flow > Loopモジュールを選択します。
ConditionでForeverを選択します。無限ループです。



Sound Sensor
Compare
Value
1

15 Sound Sensor

Sensor > Sound Sensorモジュールを選択します。
Compare : >を選択します。指定した値より大きい時を意味します。
Value : 1に設定します。音の中央値は0です。ここでは音の入力感度を下げて1よりも大きかった場合、つまり、確実に右から入力を受けた場合にのみ反応するように設定します。周囲の雑音の入力や、両手を一緒に上げる現象を防止するためです。

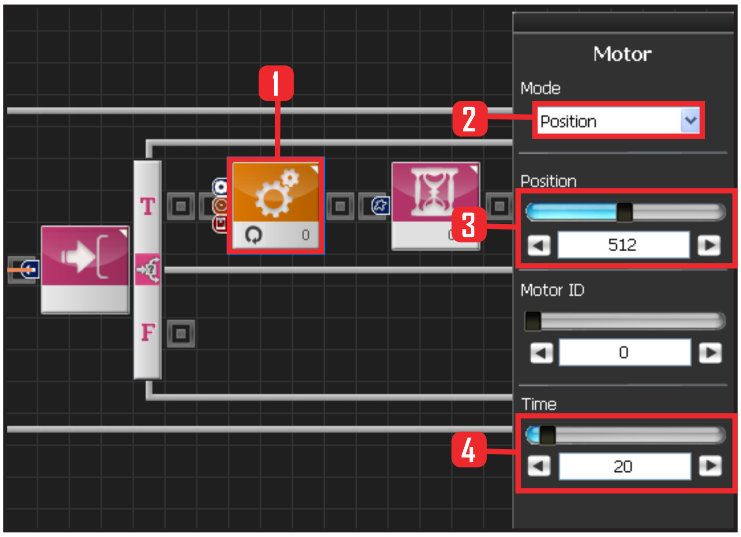

Switch

Input

☐ True ☐ False

16 Switch IF 分岐文

上記の条件がTrueの場合にのみ実行させせます。


Motor

Mode: Position

Position: 512

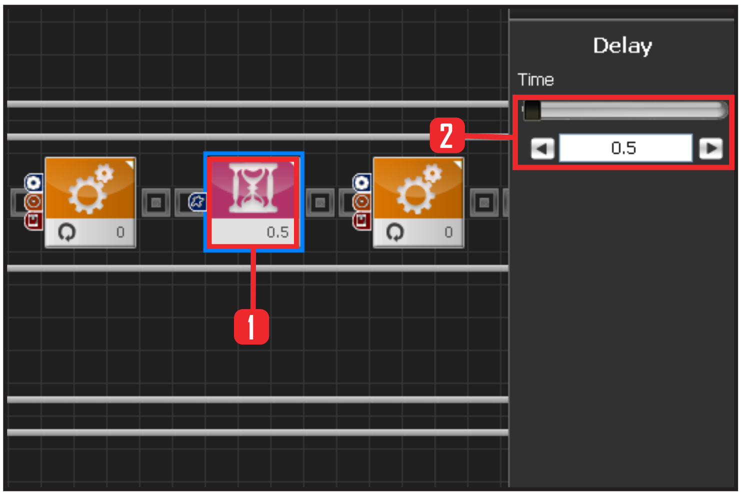
Motor ID: 0

Time: 20

17 モーター0番(右肩)の設定

音の方向の値が1より大きい場合(True)は右腕を上げます。
腕が動作している時に外部からのノイズやモーター音によって、他の腕が動作する場合があります。これを防止するために腕を動作させるときは他の腕が動作しないようにプログラミングします。

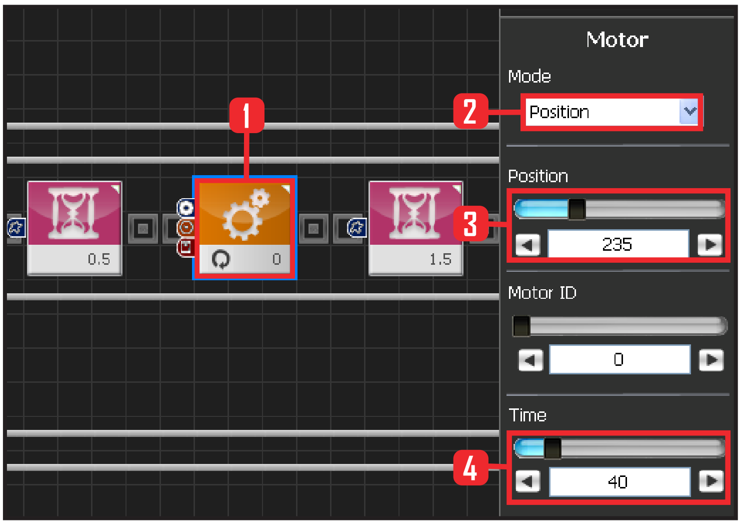
Motion > Motorを選択します。
Mode:Positionを選択します。
Positionは512に設定します。512は両腕を左右に広げた姿勢です。
Time時間は20に設定します。
ロボットが不動の姿勢で右腕を90度上げます。


Delay

Time: 0.5

18 Delay

腕の動作中に外部の雑音が聞こえたら、すぐに反応したり、他の腕を動かしたりする可能性があります。このような現象を防止するために現在の腕を動かしている間、他の現象が起こらないようにDelay値を与えます。
0.5秒のDelay中に右腕を動かしている時は他の動きはありません。



Motor

Mode: Position

Position: 235

Motor ID: 0

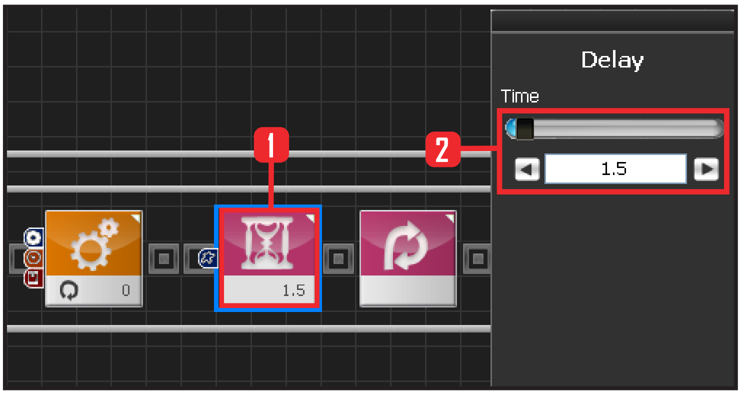
Time: 40

19 モーター0番（右肩）の復帰

90度上がった腕を再び直立不動の姿勢に戻します。

Motion > Motorを選択します。
Mode : Positionを選択します。
Positionを235にすることで不動の姿勢に戻ります。
時間は40に設定します。

ロボットの右腕が不動の姿勢に戻ります。

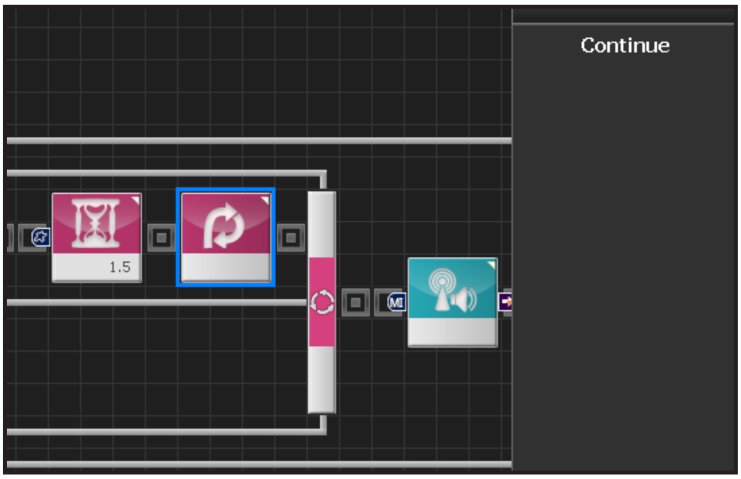


Delay

Time: 1.5

20 Delay

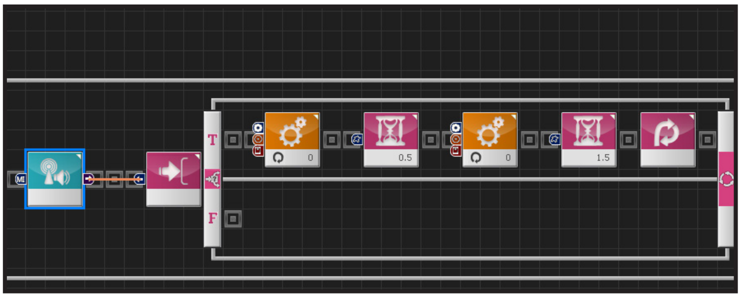
直立不動の姿勢に戻った後、しばらくの間、他の操作ができないようにDelay値を与えます。
1.5秒間のDelay値を与えると、ロボットは腕を下げた後、動作せず、音の入力も確認しません。



Continue

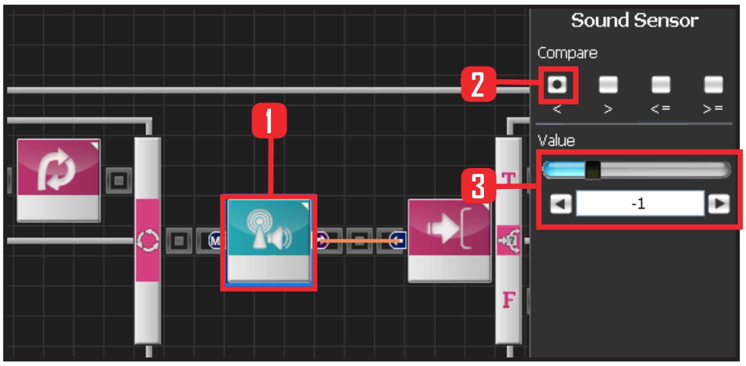
21 Continue

1.5秒間Delay後、再びLoopの最初の部分に戻ります。



22 まとめ

ロボットの右からの音に応じてロボットが正確な動作を示すために外部からの刺激を意図的に防ぐプログラミングをしました。
このプログラミングは一度の拍手で一度だけ手を上げることで精度を高めています。



Sound Sensor

Compare

● <　☐ >　☐ <=　☐ >=

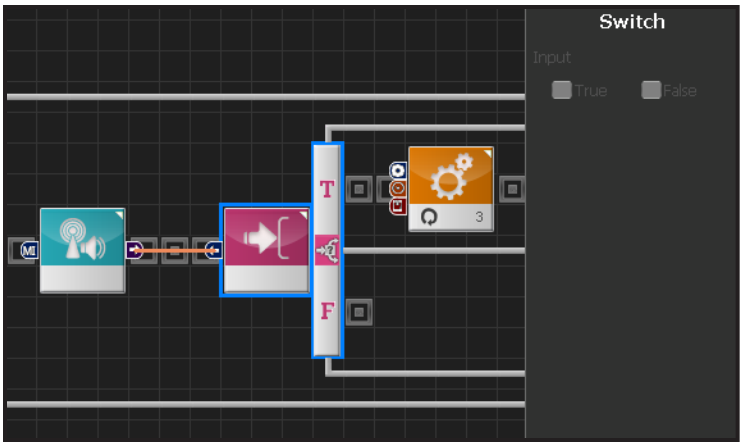
Value

-1

23 Sound Sensor（2番目）

二番目のサウンドセンサを設定します。左手が反応します。

Sensor > Sound Sensorモジュールを選択します。
Compare：<に設定します。
Valueの値は-1です。 -1よりも小さい時に反応します.



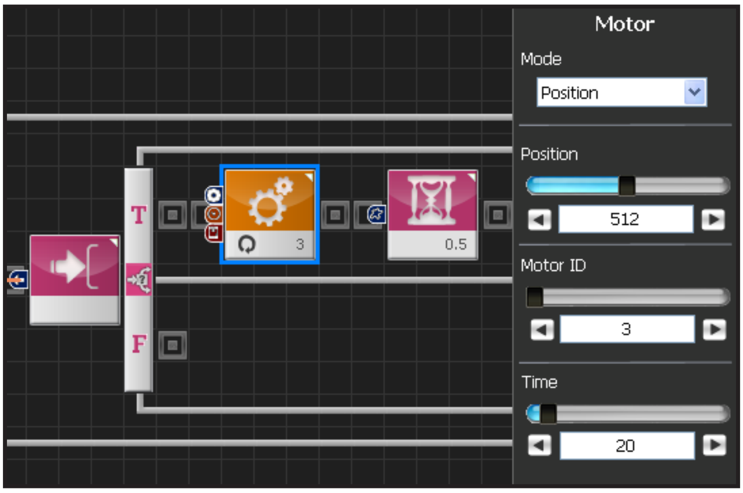
Switch

Input

☐ True　☐ False

24 Switch IF 分岐文

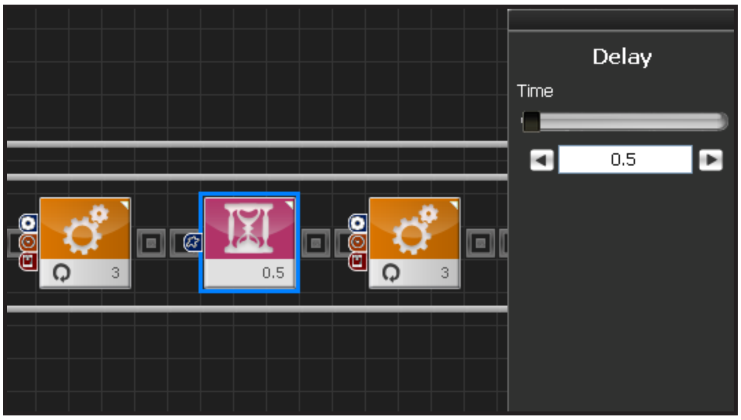
Trueの場合に動作します。


Motor
Mode: Position
Position: 512
Motor ID: 3
Time: 20

25　モーター3番（左肩）の設定

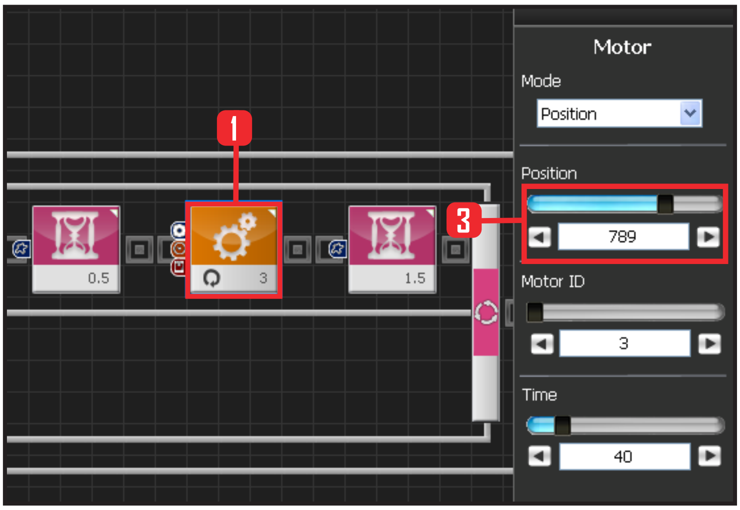
音の方向の値が-1より小さい場合（True）は左腕を上げます。

Motion > Motorを選択します。
Mode：Positionを選択します。
Positionを512に設定します。512は両腕を左右に広げた姿勢です。
Time時間は20に設定します。


Delay
Time: 0.5

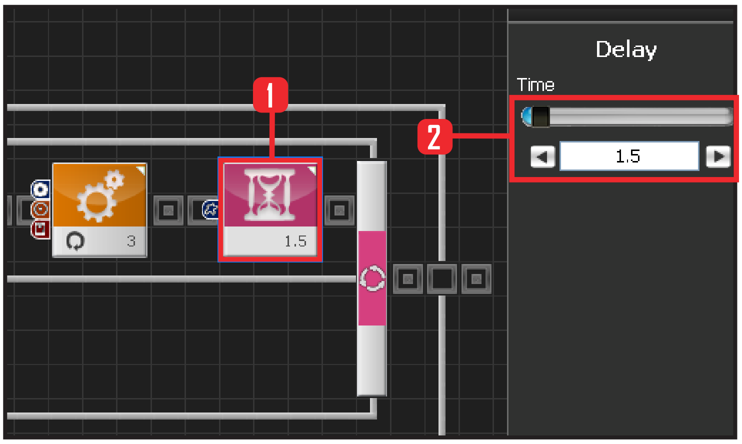
26　Delay

右腕と同じように外部の刺激（音）の認識を避けるために0.5秒間のDelay値を与えます。


Motor
Mode: Position
Position: 789
Motor ID: 3
Time: 40

27　モーター3番（左肩）の復帰

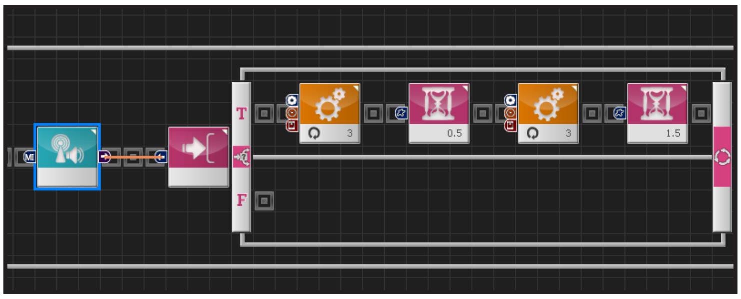
3番モーターPositionを789に設定して、再び直立不動の姿勢に戻ります。


Delay
Time
1.5
1
2
3
1.5

28 Delay

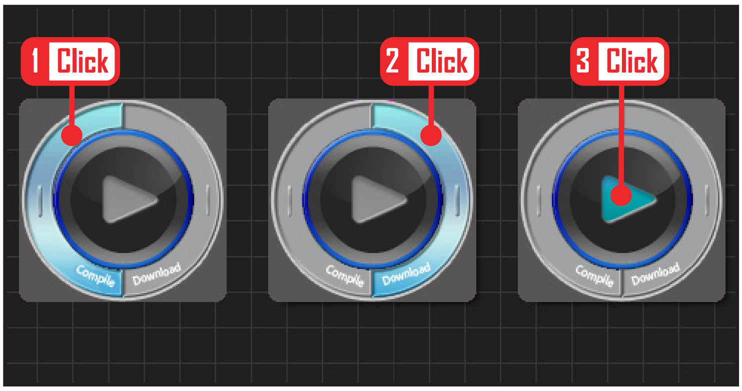
1.5秒間Delay値を与えて、他の反応が起こらないようにします。

前の0番モーターにはContinueがあるが、後ろの3番のモーターにはない理由はLoopの終わりなので、自動的に最初の部分に戻るからです。



29 左腕の反応

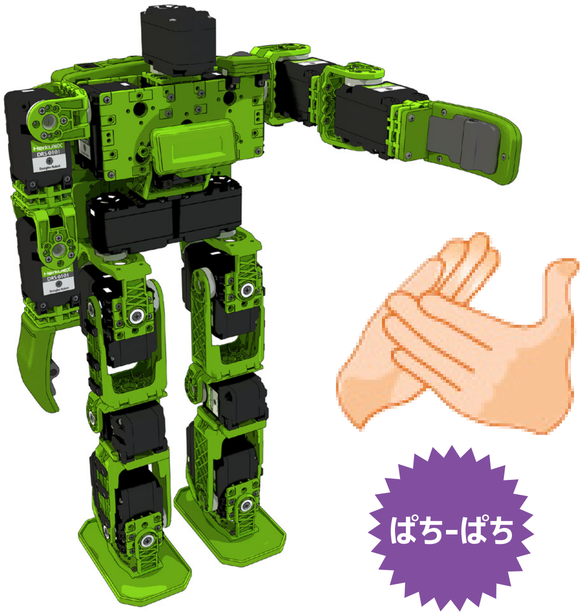
左側から拍手をした時に一定の間隔で腕を上げて広げた後、再び下げます。
拍手を連続して2回するときも、ロボットは一度だけ反応します。Delay値を与えたからです。
このようなプログラミングは周辺がうるさい時や左右の区別が難しくてロボットが混乱するときに便利に利用できます。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

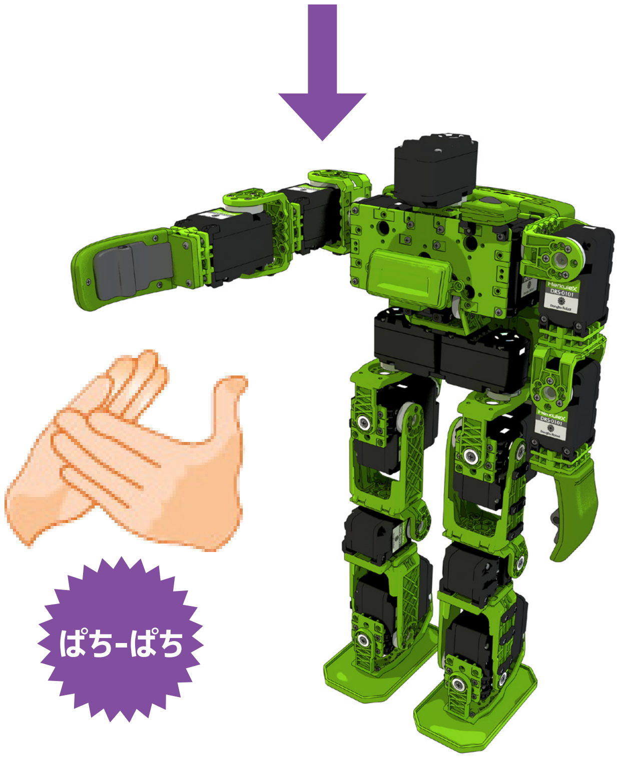
30 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。

1



2



31 ロボット動作

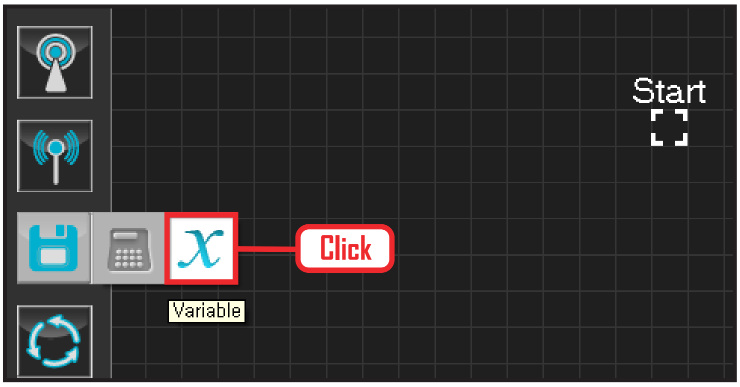
ロボットの右側から拍手する場合は右腕を上げます。ロボットの左側から拍手する場合は左腕を上げます。

Digital Distance Sensorの例

例の説明

アナログ距離センサはすべての距離測定が可能ですが、デジタル距離センサは特定の距離を基準に近いか遠いかだけを判断します。車輪付きのロボットは壁を検出するのではなく、崖の検出に多く使用されています。ヒューマノイドでは足が動くので、崖の検出ではなく壁を検出するプログラミングをします。アナログ距離センサと比較してください。

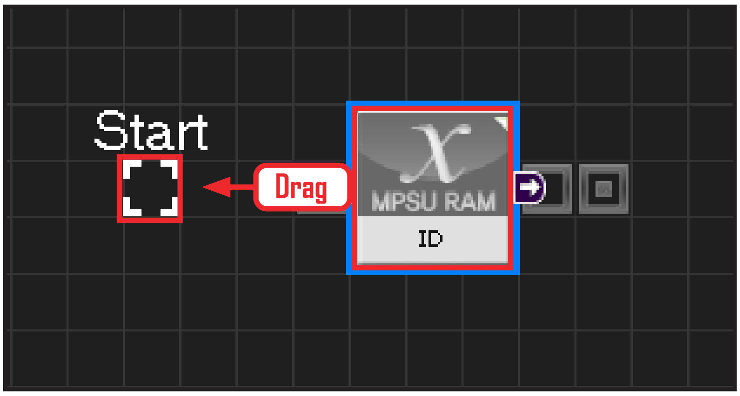
壁が一定距離に近づくと、後ずさりしてから方向を変えて前進するプログラムです。この例を実行する場合はADCポート1番（左）に、デジタル距離センサを装着する必要があります。



01 変数の指定

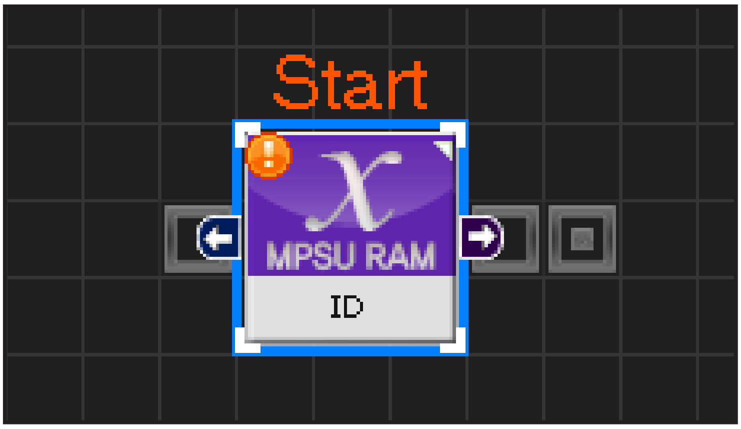
ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data > Variableモジュールをクリックします。



02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。



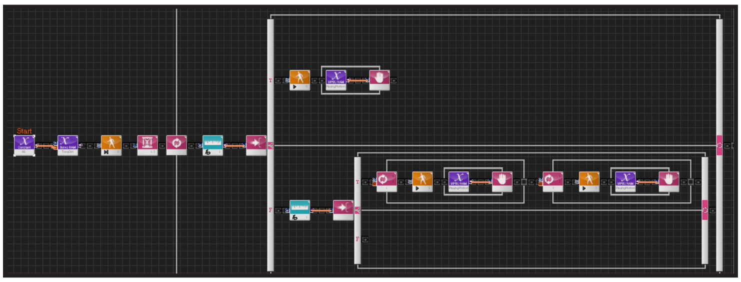
03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。

これはプログラミングが開始されたという意味です。

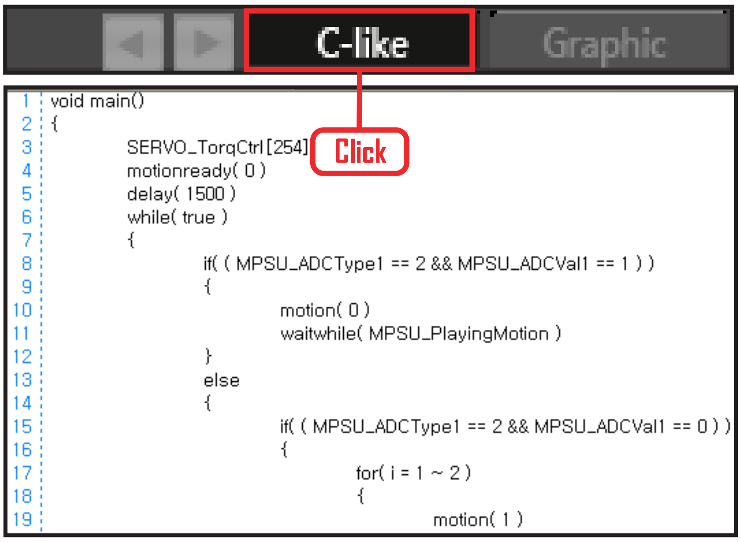
04 完全プログラミング

デジタル距離センサを利用した完全なプログラムです。



05 C-Like

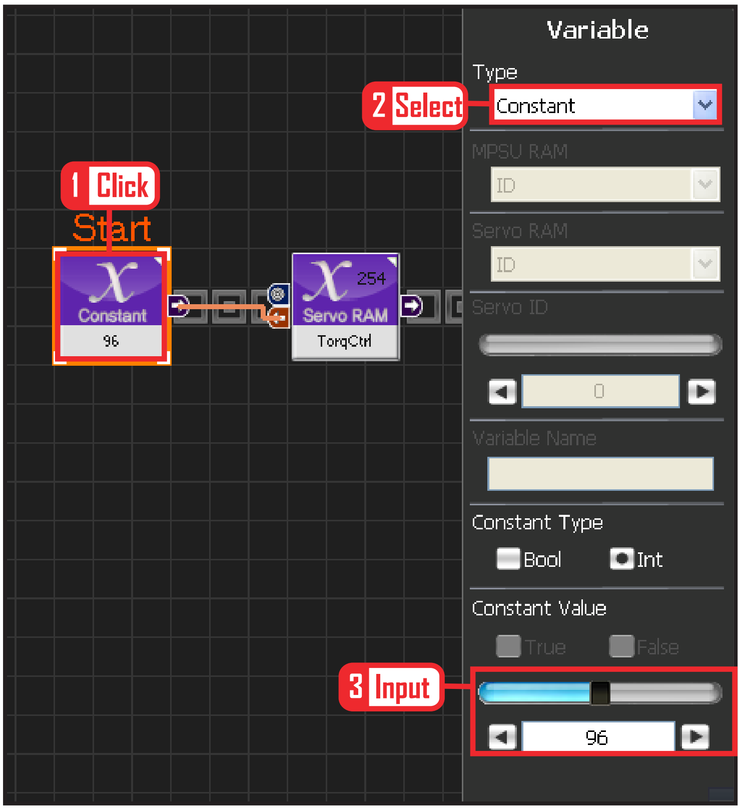
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
デジタル距離センサを利用して完全なプログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。



```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 1 ) )
                {
                        motion( 0 )
                        waitwhile( MPSU_PlayingMotion )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 0 ) )
                        {
                                for( i = 1 ~ 2 )
                                {
                                        motion( 1 )
```

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かせることができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。



Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

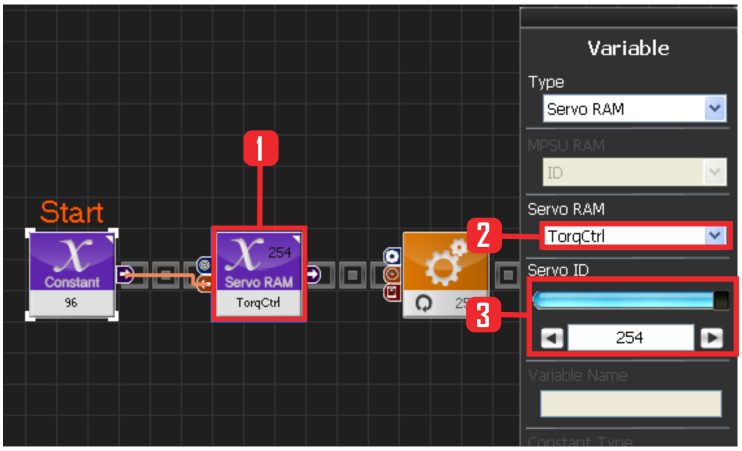
Servo ID: 0

Variable Name:

Constant Type: ○ Bool ● Int

Constant Value: ○ True ○ False

96

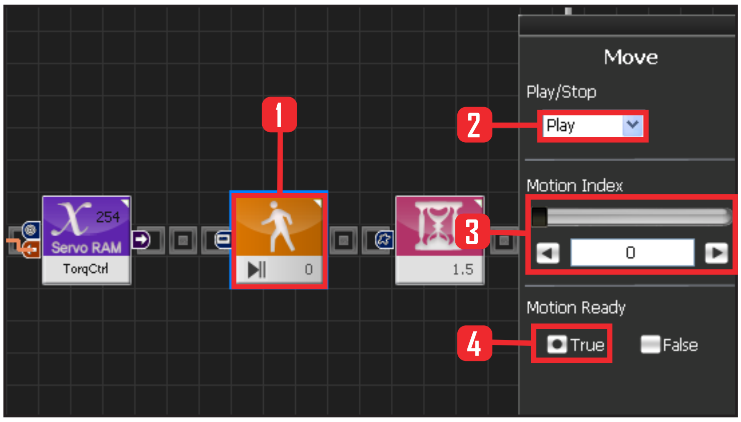

Start
Constant
96
Servo RAM
254
TorqCtrl
1
2
3
Variable

Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name
Constant Type

07 すべてのサーボに適用

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type:Servo RAMを選択します。
Servo RAM:TorqCtrlを選択します。
Servo　ID:254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

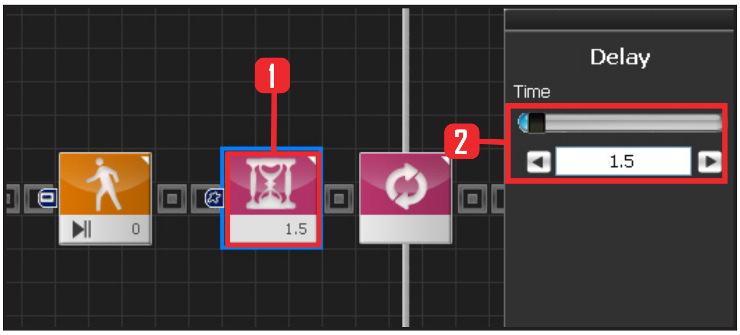

Servo RAM
254
TorqCtrl
0
1.5
1
2
3
4
Move

Play/Stop: Play
Motion Index: 0
Motion Ready: ● True ○ False

08 モーション動作の準備

モーション動作のための準備過程です。ロボットのモーションが急に変動した場合、ロボットに無理がかかります。したがって、現在実行しようとするロボットの最初のモーションからゆっくりと移動させます。Motion　ReadyがTrueの場合、モーションの最初のシーンを準備し、Falseの場合モーションが動作します。

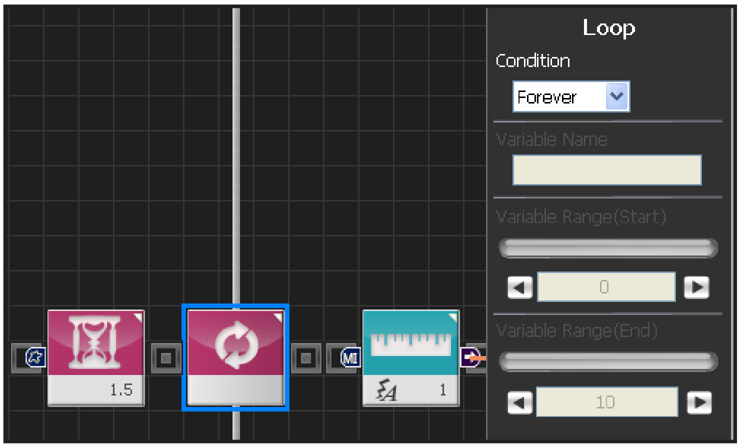
Motion > Moveモジュールを選択します。
Play/ Stop:Playを選択します。
Motion Index:0番を選択します。0番は前進しながら歩くモーションです。
Motion Ready:Trueを選択します。


0
1.5
1
2
Delay

Time: 1.5

09 Delay

MotionReady動作が終了する前にモーションが進行することを防止するためにDelay値を1.5秒に設定します。



Loop

Condition

Forever

Variable Name

Variable Range(Start)

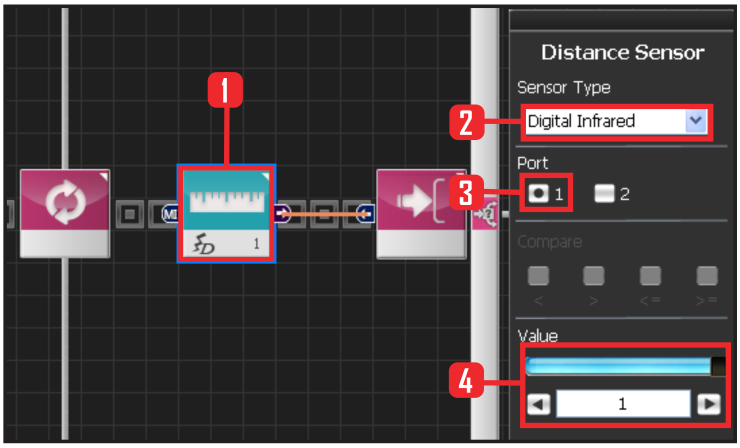
0

Variable Range(End)

10

10 反復

Forever無限反復を設定します。



Distance Sensor

Sensor Type

Digital Infrared

Port

● 1 ☐ 2

Compare

☐ < ☐ > ☐ <= ☐ >=

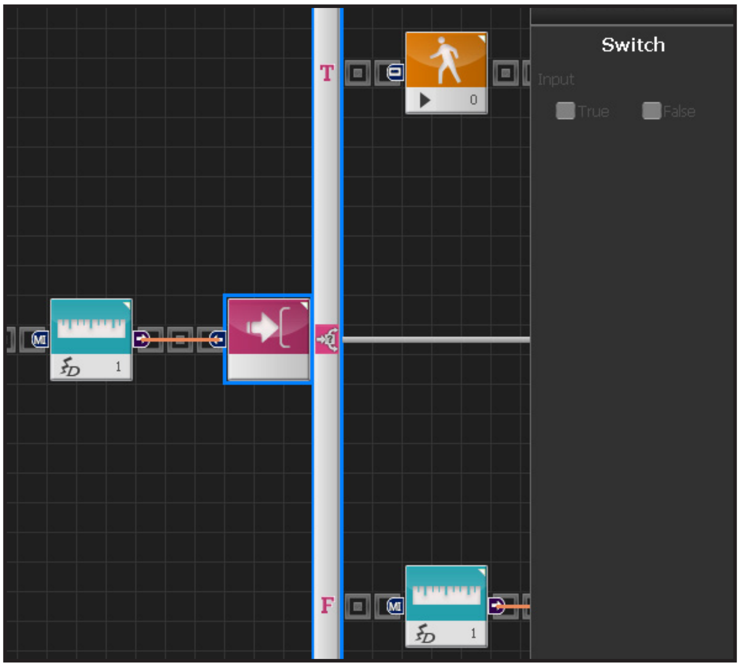
Value

1

11 デジタル距離センサの設定

デジタル距離センサは製品ごとに測定距離が異なります。ここでは20cm単位で設定されています。

Sensor > Distance Sensorモジュールを選択します。
Sensor Type : Digital Infraredを選択します。
Port : 1番を選択します。
Value : 1を選択します。 10cm以上離れていることを意味します。



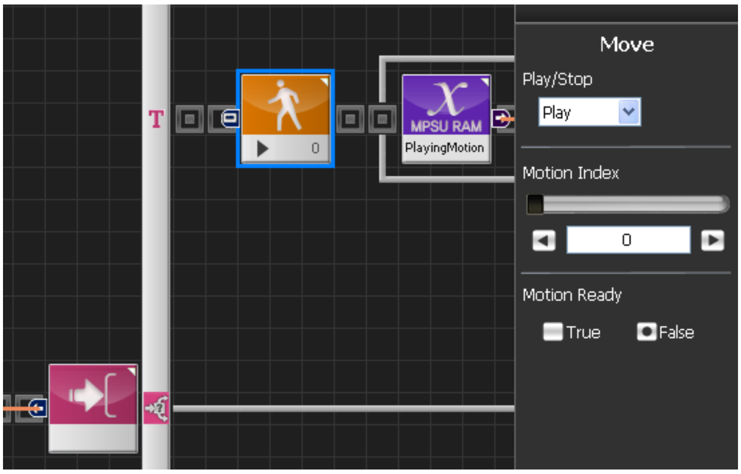
Switch

Input

☐ True ☐ False

12 If 分岐文

Trueの場合は前進し、Falseの場合は次の条件文に入ります。



Move

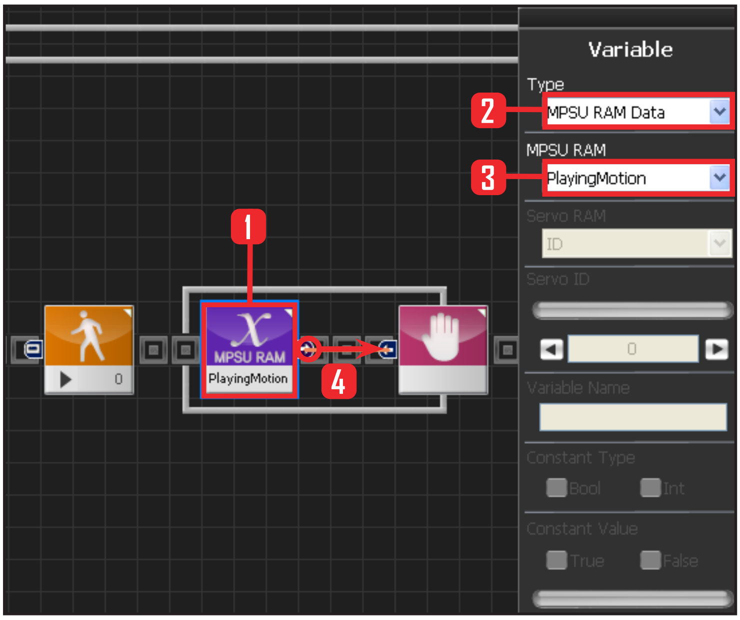
Play/Stop: Play

Motion Index: 0

Motion Ready: ○True ●False

13 前進する

10cmより遠い距離なので、ロボットが前進します。Motion Ready値でFalseを選択すると、ロボットは前進モーションを実行します。



Variable

Type: MPSU RAM Data

MPSU RAM: PlayingMotion

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type: ○Bool ○Int

Constant Value: ○True ○False

14 モーション動作の確認

Loopは継続的な繰り返しを意味します。Move命令を下してから、実際のモーションが実行され、完了するまでは時間がかかるので、loop内にMoveモジュールの一つだけを入れて実行すると、モーションが既に実行中央にもかかわらずloopを続けてモーションを実行する命令を繰り返すことになります。

この場合、Moveモジュールを実行した回数と実際のモーションを実行する回数が異なります。

したがって実行したモーションが終わるまで待ってから再びloopの最初に戻す方がより確実です。

Variable > MPSU RAM Dataに入ると、Playing Motion があります。Playing Motionはロボットのモーションを実行しているかを確認する変数です。そのPlaying Motionに Waitをかければ、ロボットの動作が終わるまでLoopを待ちます。

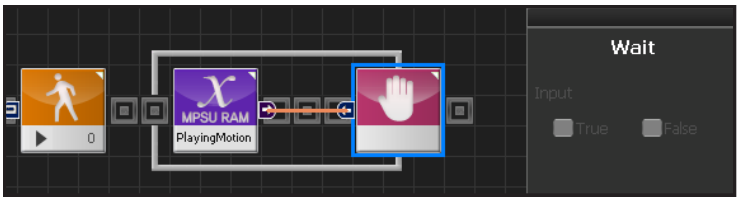
Data > Variableモジュールを選択します。

Type：MPSU RAM Dataを選択します。

MPSU RAM：Playing Motionを選択します。

Outputコネクタの値の後ろのWaitモジュールに接続します。

モーションが終わるまで待つことを意味します。



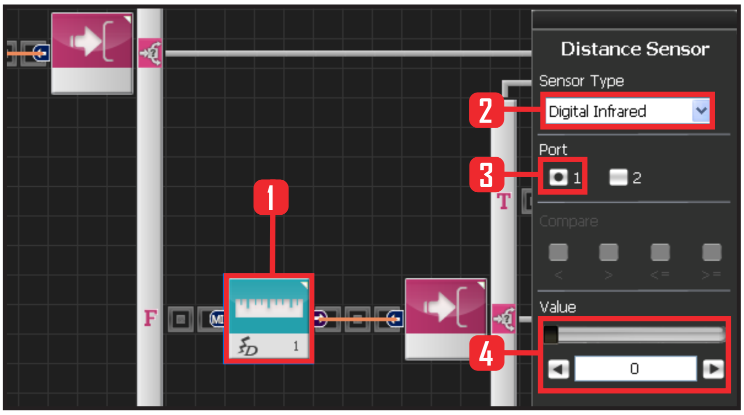
Wait

Input: ○True ○False

15 Wait

モーションが終わるまで待ちます。

モーションが終わればまた最初に戻ってモーションを繰り返します。

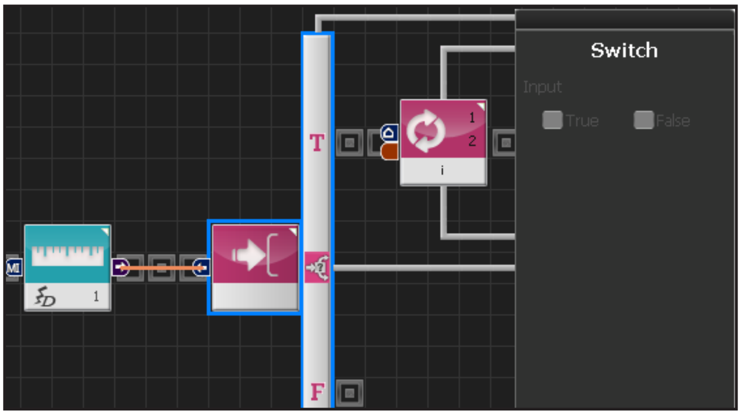


Distance Sensor
Sensor Type: Digital Infrared
Port: ● 1 ☐ 2
Compare
Value: 0

16 壁に接近する時の動作

ロボットが壁の10cm以内に接近した時はロボットを後進歩行させ、その後、右折させるプログラムです。

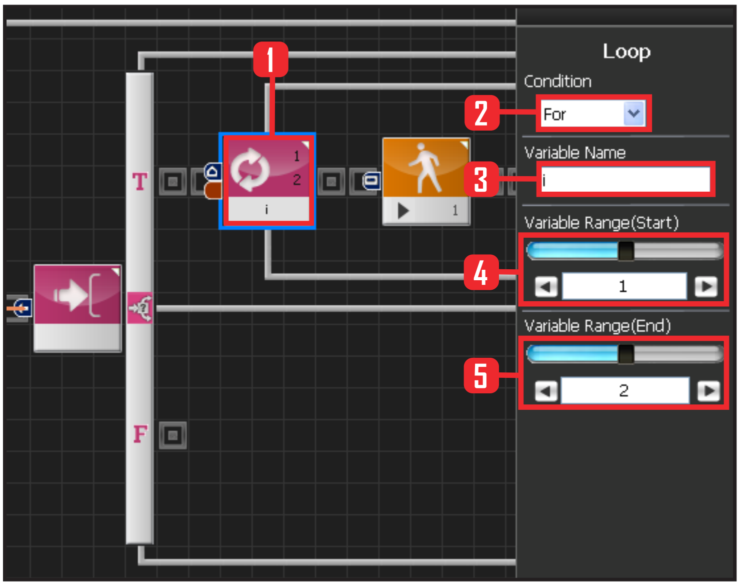
Sensor > Distance Sensorモジュールを選択します。
Sensor Type : Digital Infraredを選択します。
Port : 1番を選択します。
Value : 0を選択します。 10cm内に入ったことを意味します。



Switch
Input
☐ True ☐ False

17 If 分岐文

10cmより近いと、Trueの中の文を実行します。

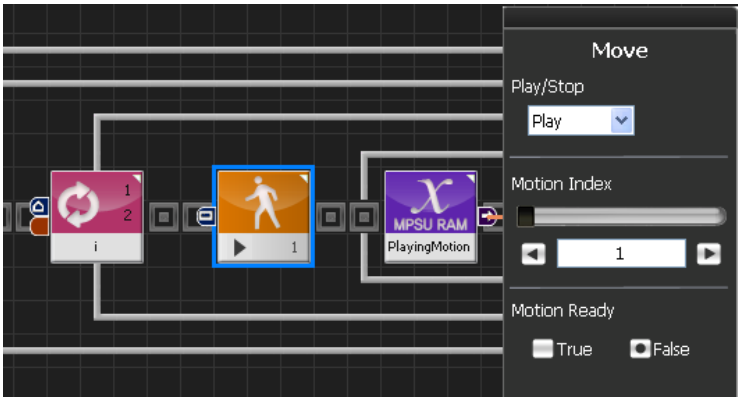


Loop
Condition: For
Variable Name: i
Variable Range(Start): 1
Variable Range(End): 2

18 For 反復

ロボットのモーションは同じ動作を繰り返して、必要なモーションまでの動作を取ることができます。
モーション1番はロボットが後進するモーションです。後進は左足と右足を一度だけ後ろに移動させることです。
そのモーションに必要な数だけ繰り返しを与えることができるFor文を用いて、移動したい位置まで移動させることができます。

Flow > Loopモジュールを選択します。
Condition : Forを選択します。
Valuable Nameに i を入力します。
Valuable Range(Start)に1を入力します。
Valuable Range(End)に2を入力します。
モーションを2回実行させるという意味です。


i
1
2
1
MPSU RAM
PlayingMotion

Move

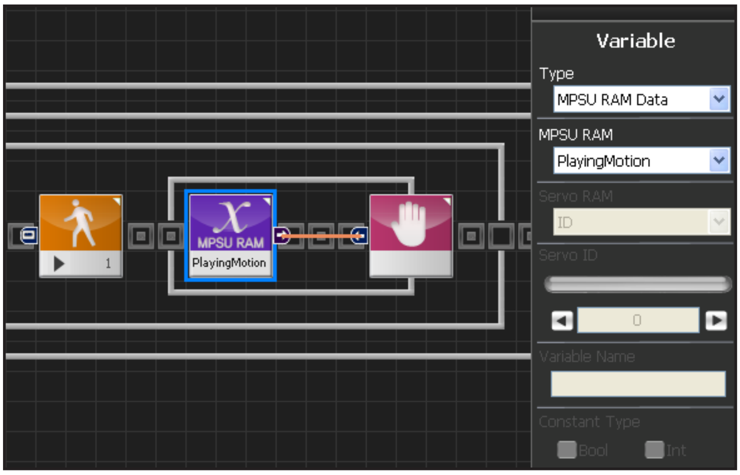
Play/Stop: Play

Motion Index: 1

Motion Ready: ☐ True ☑ False

19 後ろにしりぞく

モーション1番はロボットが後進するモーションです。
Falseを選択すると、ロボットが後進モーションを実行します。


1
MPSU RAM
PlayingMotion

Variable

Type: MPSU RAM Data

MPSU RAM: PlayingMotion

Servo RAM: ID

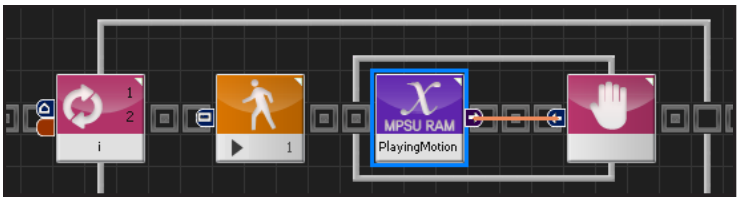
Servo ID: 0

Variable Name

Constant Type: ☐ Bool ☐ Int

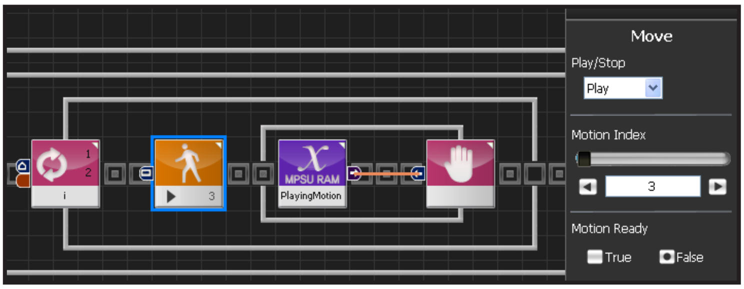
20 動作感知

Playing Motionでロボットの動作を検出し、終わればまたfor文の最初の部分に戻ります。


1
2
i
1
MPSU RAM
PlayingMotion

21 二回反復の後進

ロボットを二回後進させるプログラムです。


1
2
i
3
MPSU RAM
PlayingMotion

Move

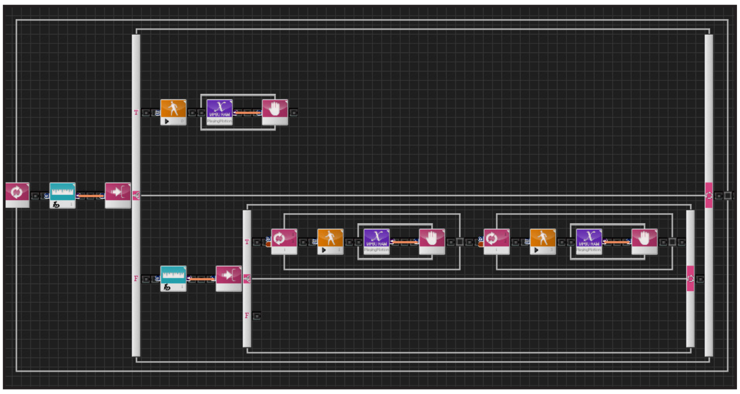
Play/Stop: Play

Motion Index: 3

Motion Ready: ○ True ● False

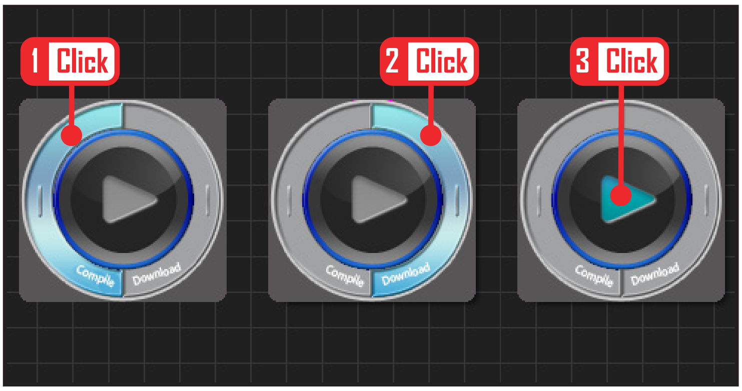
22 右折させる

3番モーションはその場で左折する動作です。右折動作も同じようにFor文を使って調整することができます。Motionで3番を選択して、For文を1～3まで設定して、上記のようにプログラムします。



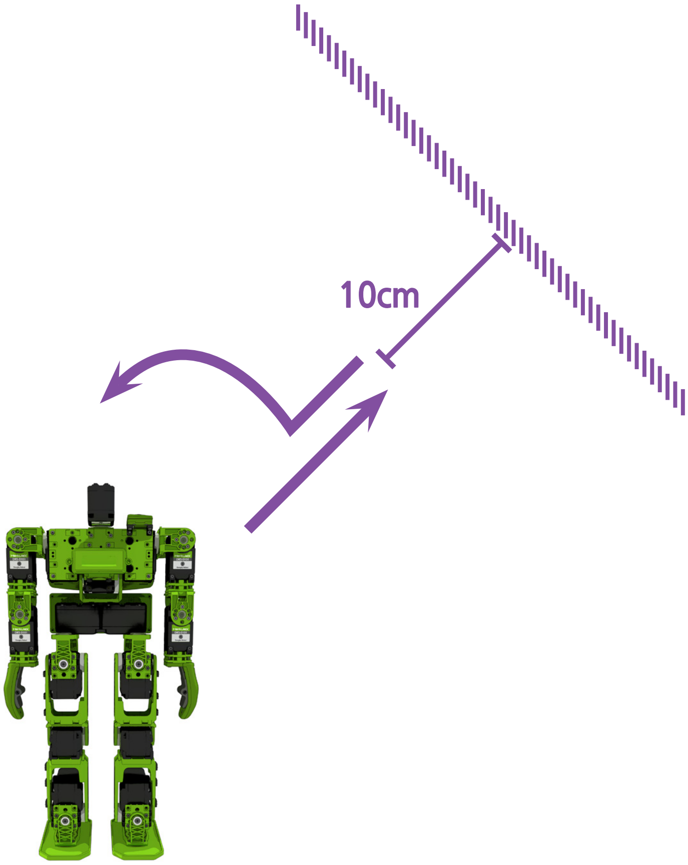
23 全体を見る

10cmよりも遠い場合にはロボットをそのまま前進させ、10cmより近い場合には、For文で後進と右に曲がるモーションを絶えず繰り返して壁を回避するプログラミングです。


1 Click
2 Click
3 Click
Compile
Download

24 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。



25 ロボット動作

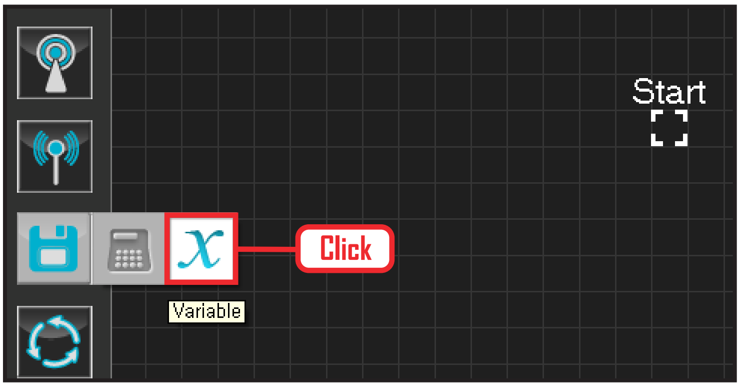
ロボットは10cm前の壁を検出すると、後進してから右折し、そのまま前進します。

Analog Distance Sensorの例

例の説明

アナログ距離センサを用いて壁に近づくと左折して壁を回避するプログラムです

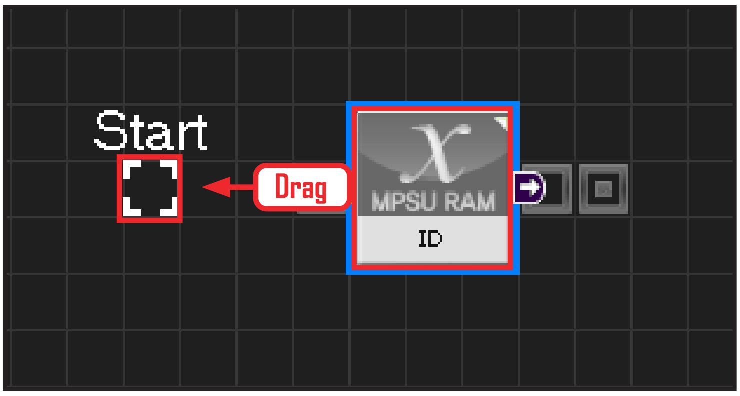
距離センサはアナログセンサとデジタルセンサがあります。デジタルセンサは一定の距離（10cm）を基準に内側と外側の両方の領域のみを検出するのに対しアナログセンサは距離別に距離（6～80cm）を検出することができます。この例を実行するときはADCポート1番（左）にPSDセンサを装着する必要があります。



01 変数の指定

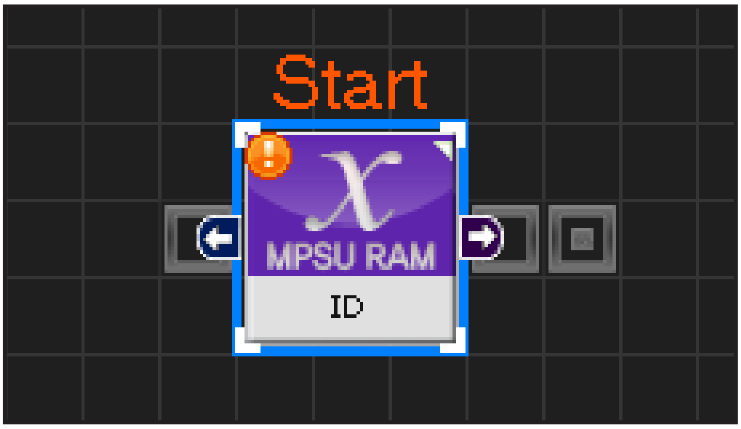
ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。

Data ＞ Variableモジュールをクリックします。



02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。



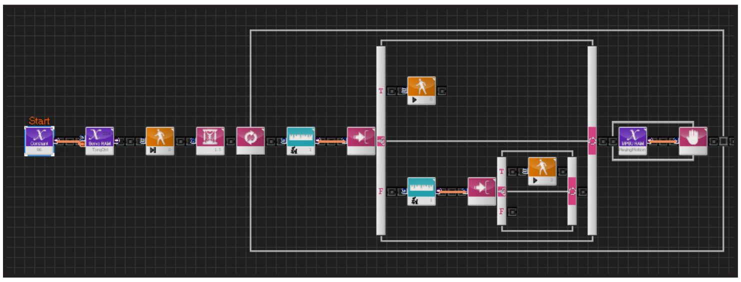
03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。

これはプログラミングが開始されたという意味です。

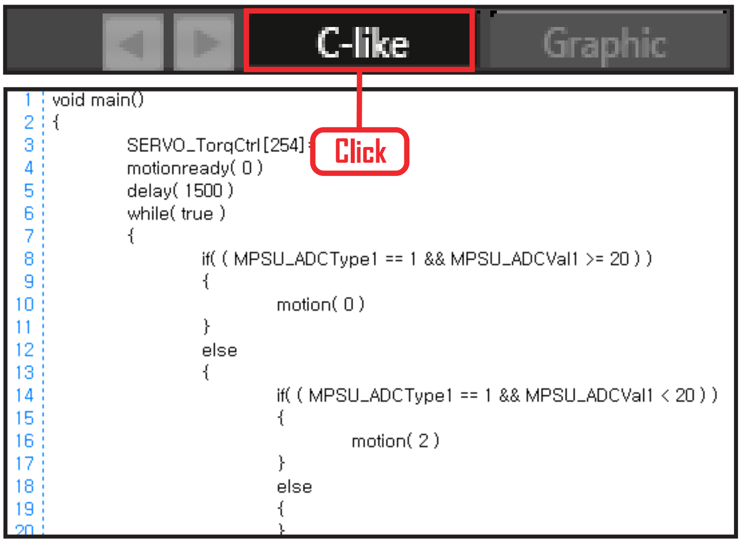
04 完全プログラミング

アナログ距離センサを利用した完全なプログラムです。



05 C-Like

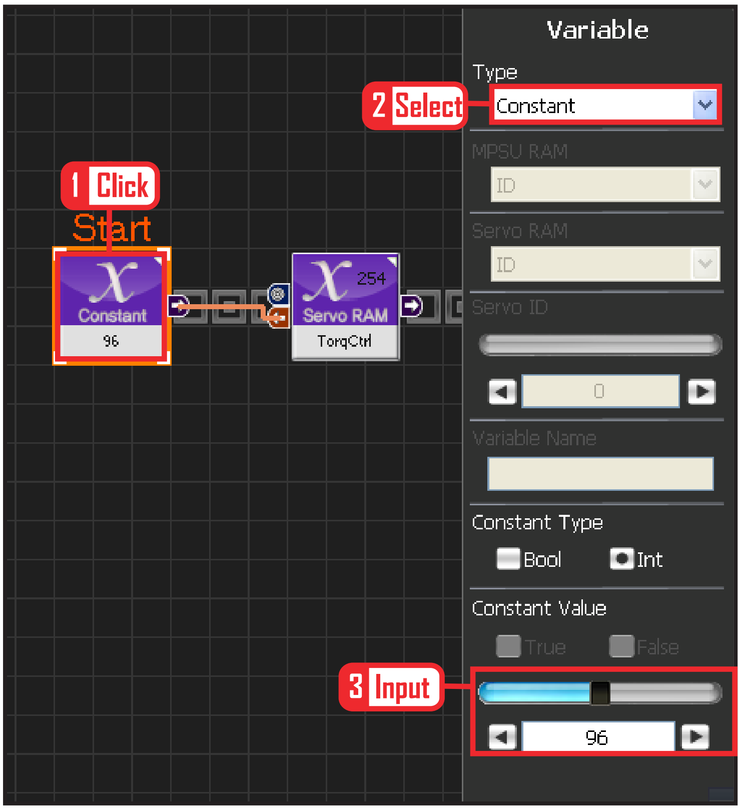
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
アナログ距離センサを利用して完全なプログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。


C-like
Graphic
Click

```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 1 && MPSU_ADCVal1 >= 20 ) )
                {
                        motion( 0 )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 1 && MPSU_ADCVal1 < 20 ) )
                        {
                                motion( 2 )
                        }
                        else
                        {
                        }
```

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant　Value値を96に設定します。サーボのTorqControlレジスタに96（0x60）という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。

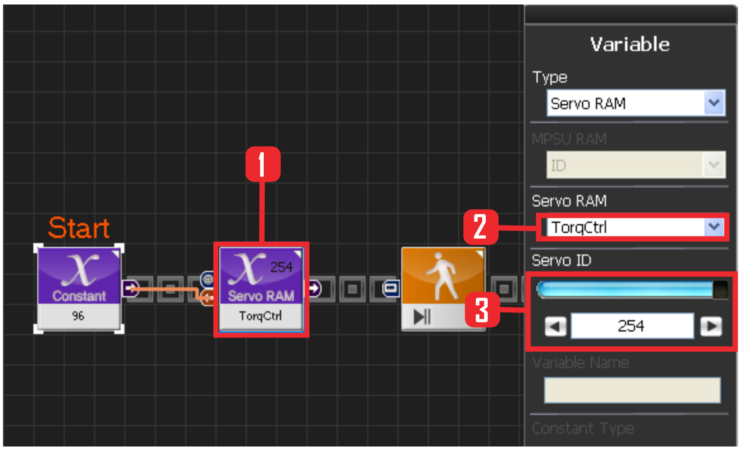

1 Click
Start
Constant
96
254
Servo RAM
TorqCtrl
2 Select
3 Input
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Value
True
False

Type: Constant

Constant Type: ○ Bool ● Int

Constant Value: 96

07 すべてのサーボに適用


Start
Constant
96
254
Servo RAM
TorqCtrl
1
2
3
MPSU RAM
ID
Variable Name
Constant Type

Variable

Type: Servo RAM

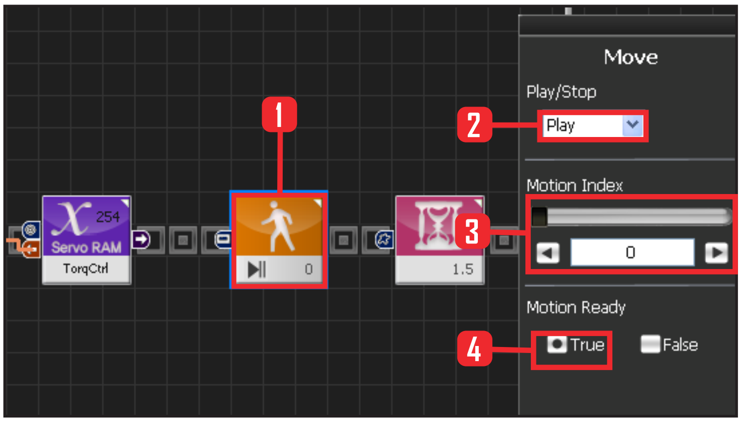
Servo RAM: TorqCtrl

Servo ID: 254

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type : Servo RAMを選択します。
Servo RAM : TorqCtrlを選択します。
Servo　ID : 254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

08 モーション動作の準備


254
Servo RAM
TorqCtrl
0
1.5
1
2
3
4

Move

Play/Stop: Play

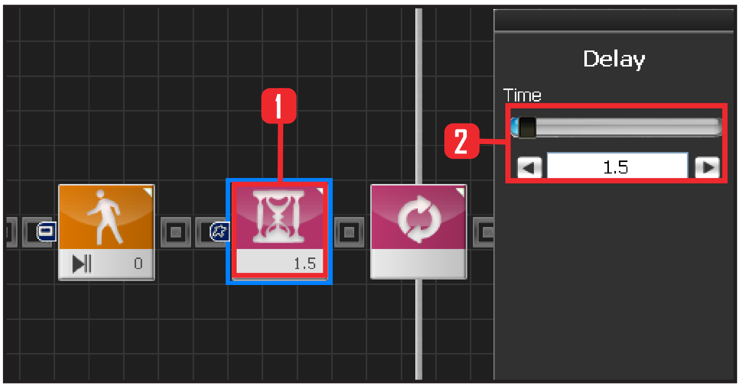
Motion Index: 0

Motion Ready: ● True ○ False

モーション動作のための準備過程です。ロボットのモーションが急に変動した場合、ロボットに無理がかかります。したがって、現在実行しようとするロボットの最初のモーションからゆっくりと移動させます。Motion　ReadyがTrueの場合、モーションの最初のシーンを準備し、Falseの場合モーションが動作します。

Motion > Moveモジュールを選択します。
Play/ Stop : Playを選択します。
Motion Index : 0番を選択します。0番は前進しながら歩くモーションです。
Motion Ready : Trueを選択します。

09 Delay



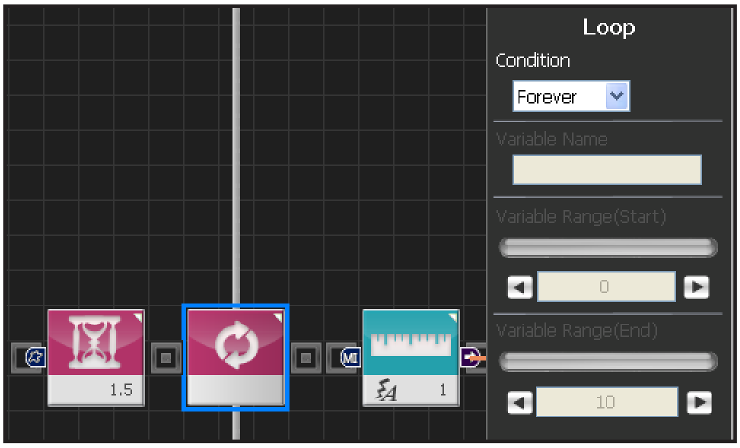
Delay

Time: 1.5

MotionReady動作が終了する前にモーションが進行することを防止するためにDelay値を1.5秒に設定します。

10 反復

Forever無限反復を設定します。



Loop

Condition: Forever

Variable Name

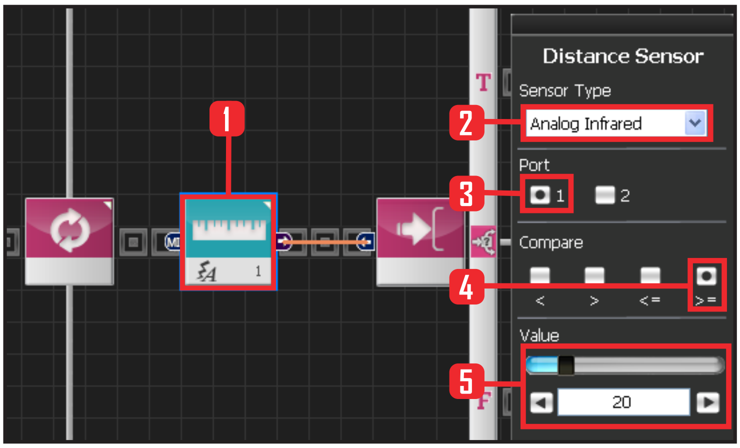
Variable Range(Start): 0

Variable Range(End): 10

11 アナログ距離センサの設定

20cm基準に設定します。

Sensor > Distance Sensorモジュールを選択します。
Sensor Type : Analog Infraredを選択します。
Port : 1番を選択します。
Compare : >= を選択します。基準よりも同じか高い時はTrueです。
Value : 20を選択します。20cmを意味します。



Distance Sensor

Sensor Type: Analog Infrared

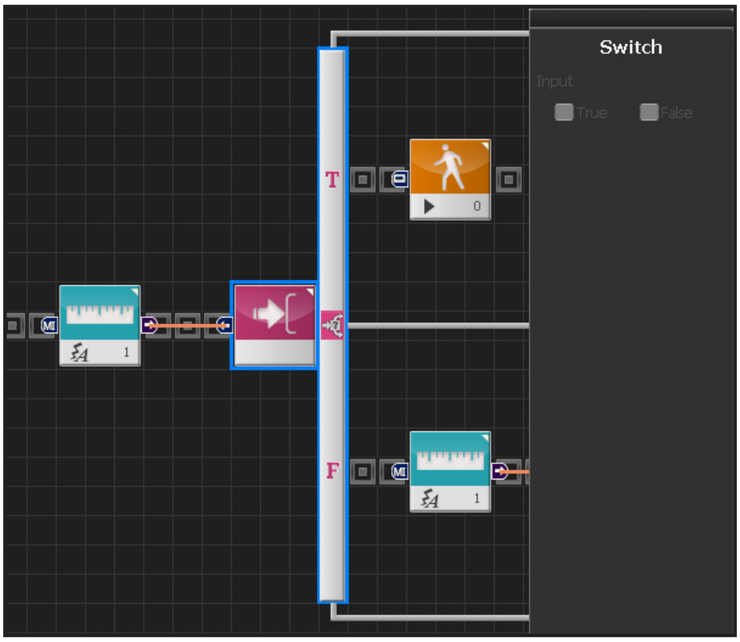
Port: ● 1 ☐ 2

Compare: ☐ < ☐ > ☐ <= ● >=

Value: 20

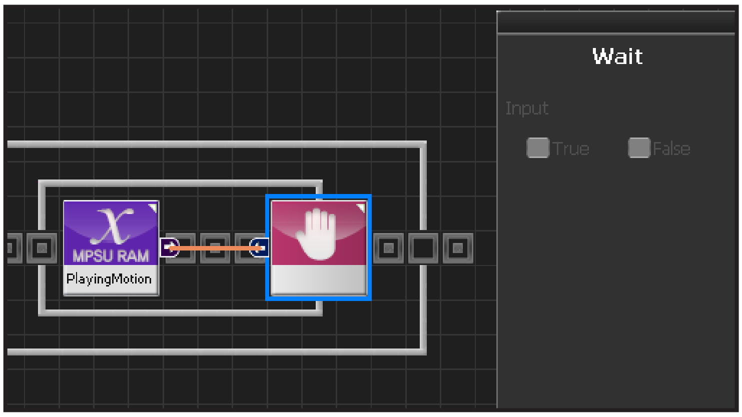
12 If 分岐文

Trueの場合は前進し、Falseの場合は次の条件文に入ります。



Switch

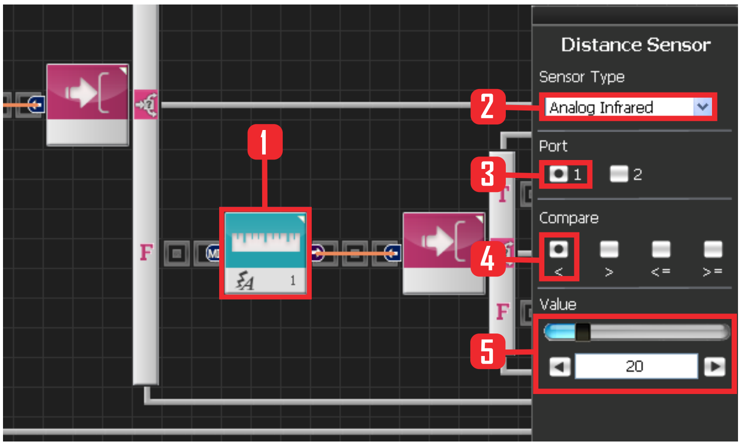
Input: ☐ True ☐ False



Wait
Input
☐ True ☐ False

13　前進する

20cmより遠い距離なので、ロボットが前進させます。
Motion　Ready値をFalseに選択すると、ロボットは前進モーションを実行します。

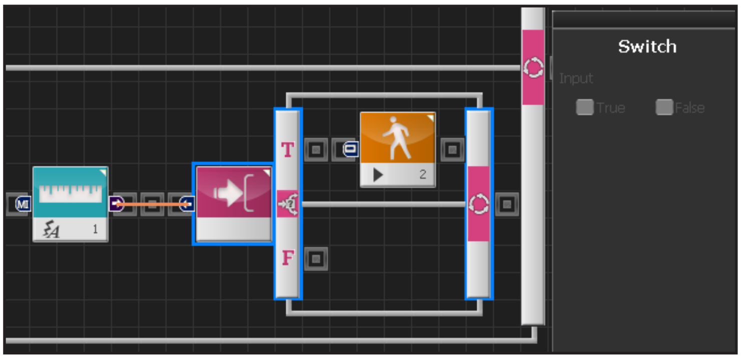


Distance Sensor
Sensor Type
Analog Infrared
Port
● 1 ☐ 2
Compare
● < ☐ > ☐ <= ☐ >=
Value
20

14　壁に接近する時の動作

ロボットが壁の20cm以内に接近した時はロボットを右折させるプログラムです。

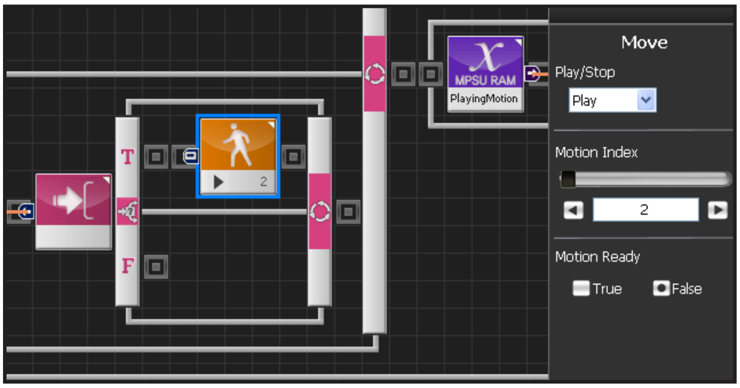
Sensor＞Distance Sensorモジュールを選択します。
Sensor Type：Digital Infraredを選択します。
Port：1番を選択します。
Compare：＜を選択します。基準より低い場合にTrueを返します。
Value：20を選択します。20cmを意味します。



Switch
Input
☐ True ☐ False

15　If 分岐文

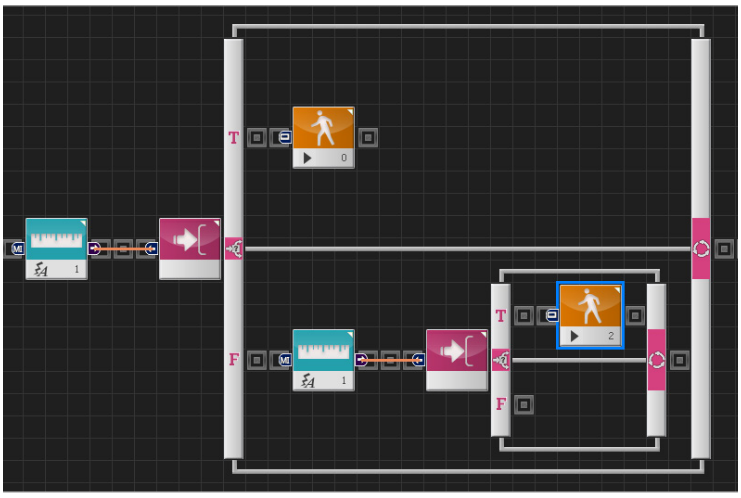
20cmより近いと、Trueの中の文を実行します。



T
F
MPSU RAM
PlayingMotion
2
Move
Play/Stop
Play
Motion Index
2
Motion Ready
True
False

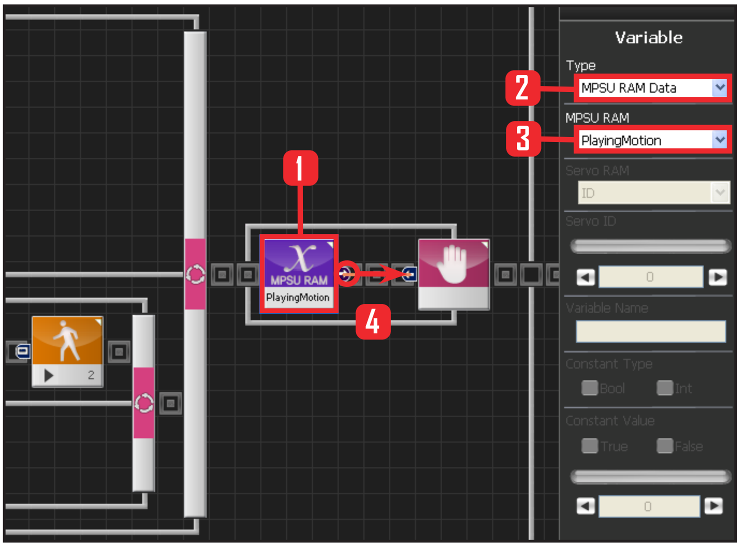
16 左折させる

モーション2番は左折モーションです。
Motion Readyの値がFalseの場合は左折モーションを実行します。



17 距離別の動作

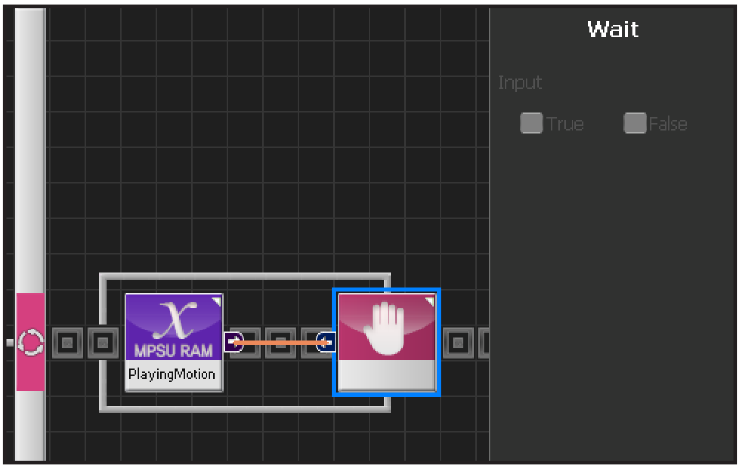
20cmより遠い場合はロボットが前に前進し、20cm未満の場合はロボットが左折します。



1
2
3
4
MPSU RAM
PlayingMotion
2
Variable
Type
MPSU RAM Data
MPSU RAM
PlayingMotion
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
Bool
Int
Constant Value
True
False
0

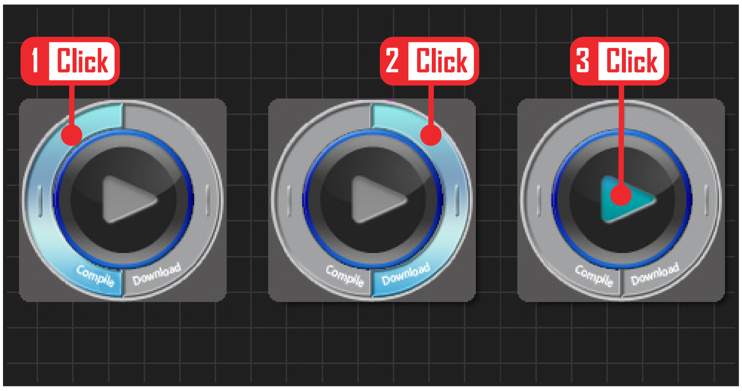
18 モーション動作の確認

Loopは継続的な繰り返しを意味します。Move命令を下してから、実際のモーションが実行され、完了するまでは時間がかかるので、loop内にMoveモジュールの一つだけを入れて実行すると、モーションが既に実行中央にもかかわらずloopを続けてモーションを実行する命令を繰り返すことになります。
この場合、Moveモジュールを実行した回数と実際のモーションを実行する回数が異なります。
したがって実行したモーションが終わるまで待ってから再びloopの最初に戻す方がより確実です。
Variable > MPSU RAM Dataに入ると、Playing Motion があります。Playing Motionはロボットのモーションを実行しているかを確認する変数です。そのPlaying MotionにWaitをかければ、ロボットの動作が終わるまでLoopを待ちます。
Data > Variableモジュールを選択します。
Type：MPSU RAM Dataを選択します。
MPSU RAM：Playing Motionを選択します。
Outputコネクタの値の後ろのWaitモジュールに接続します。
モーションが終わるまで待つことを意味します。


Wait
Input
True
False
MPSU RAM
PlayingMotion

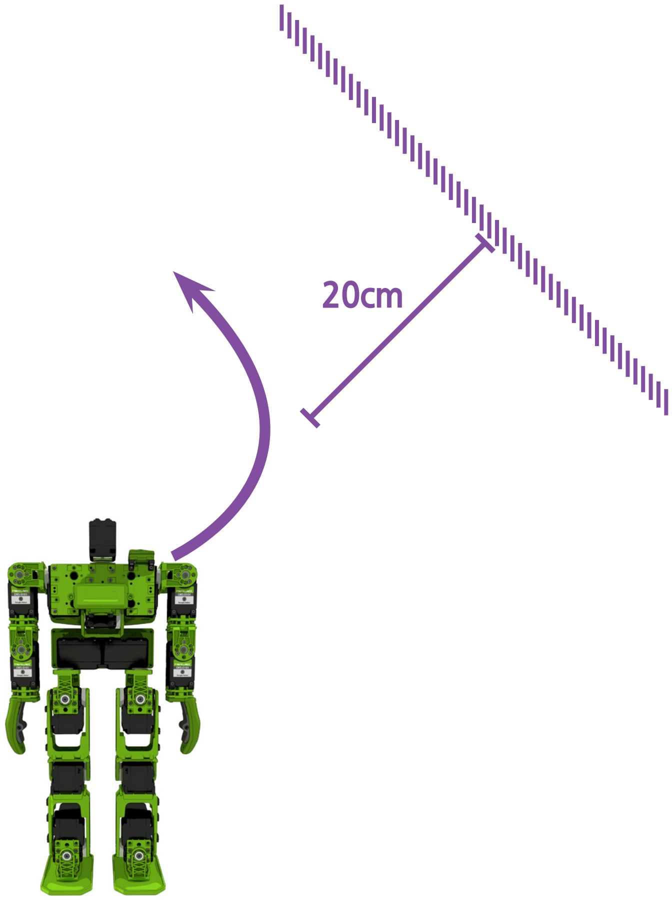
19 Wait

モーションが終わるまで待ちます。
モーションが終われぱまた最初に戻ってモーションを繰り返します。


1 Click
2 Click
3 Click
Compile
Download

20 コンパイル、ダウンロード、実行

左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。



21 ロボット動作

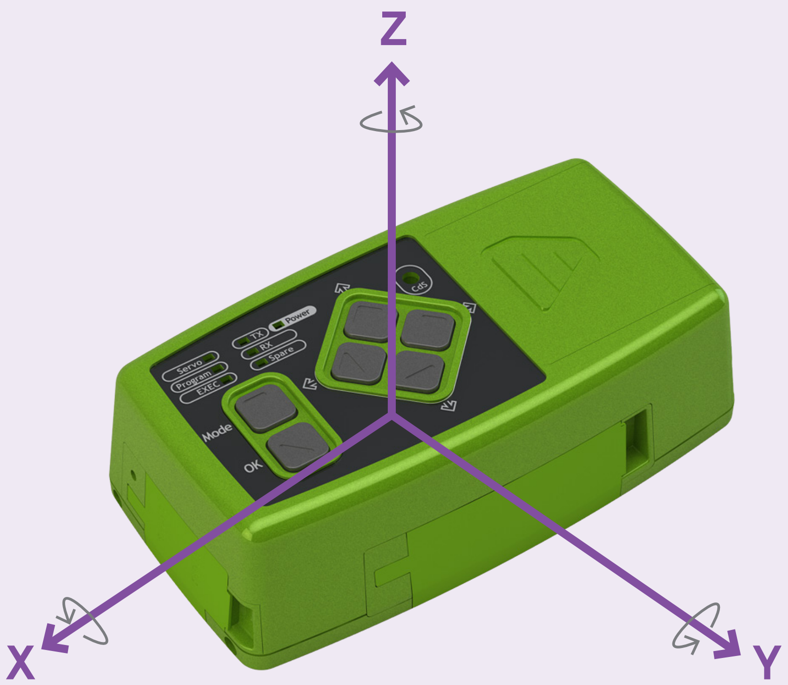
ロボットは壁に向かって前進し、20cm内に近づくと、左折します。

Acceleration Sensorの例

例の説明

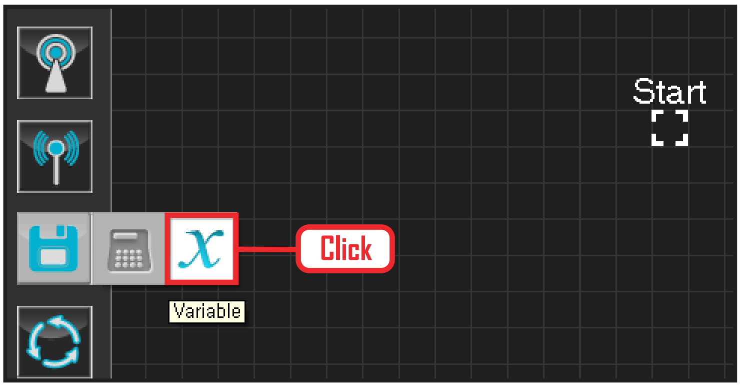
Acceleration（加速度）センサを用いてロボットが前に倒れた時と後ろに倒れた時を検出し立ち上がるプログラミングです。

Accelerationセンサはコントローラーの背面カバーを開けて入れられるモジュール形で、Gyroセンサモジュールと結合しています。


Z
X
Y
Servo
Program
EXEC
TX
RX
Power
Spare
CdS
Mode
OK

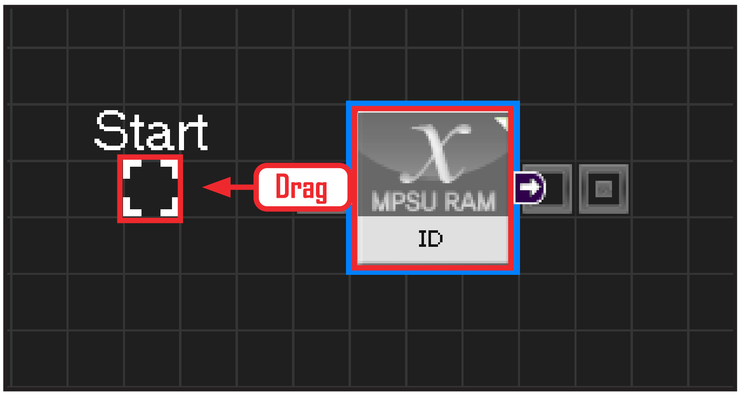
- ロボットが前に倒れた時、Z軸は“-”の加速度で、その値は約-4096です。
- ロボットが後ろに倒れた時、Z軸は“+”の加速度で、その値は約4096です。

（4096は約1gの重力の値を表します。）


Start
Click
Variable

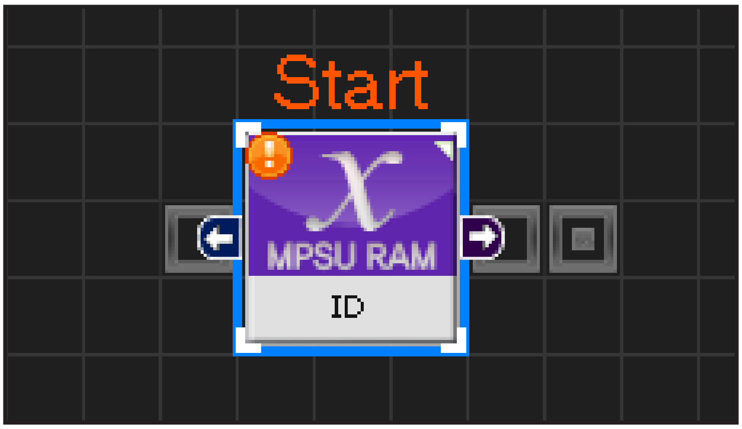
01 変数の指定

ロボットを動作させるということはロボットのサーボモーターを動作させるということです。サーボが自ら動くことができる状態の値を指定しなければなりません。
Data > Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

02 開始

モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。

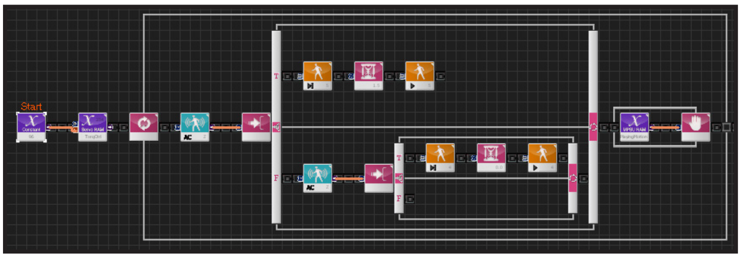

Start
MPSU RAM
ID

03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。
これはプログラミングが開始されたという意味です。

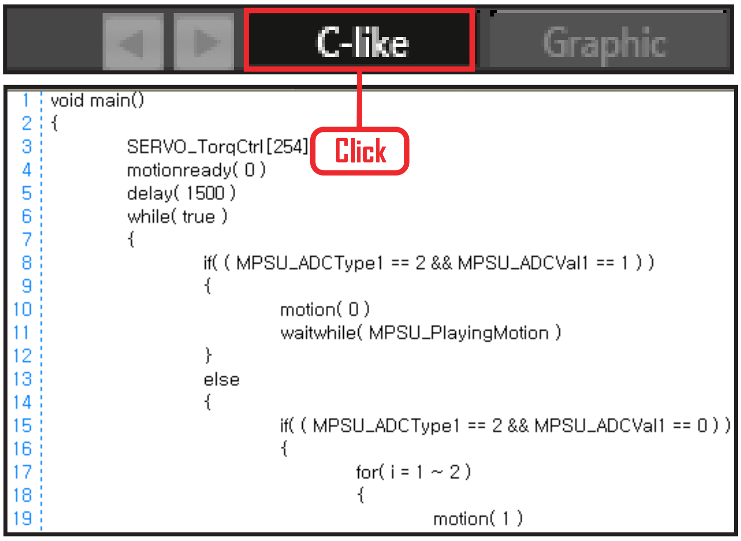
04 完全プログラミング

加速度センサを利用して倒れたロボットを立ち上げる完全なプログラムです。



05 C-Like

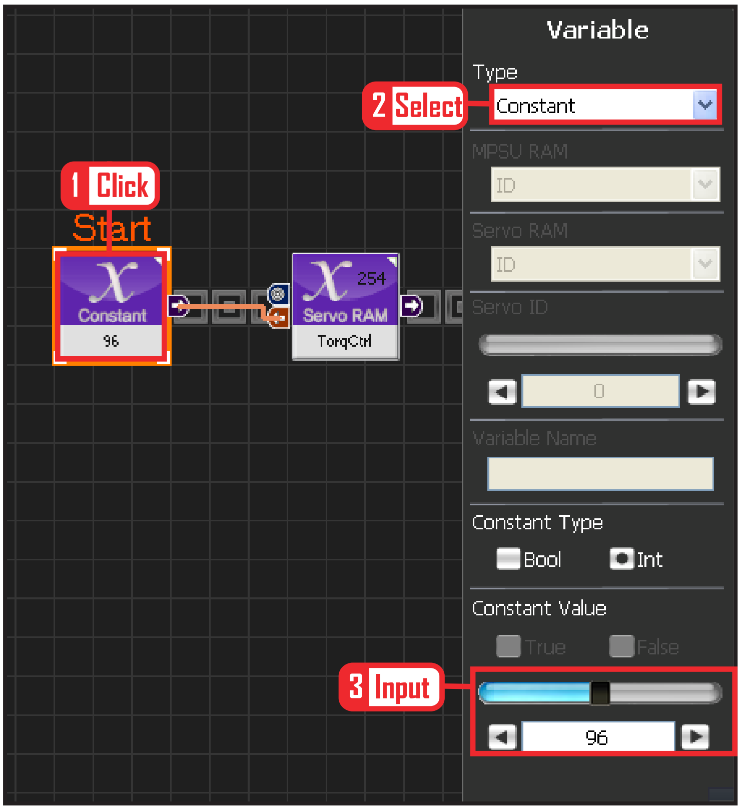
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
加速度センサを利用して完全なプログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。


C-like
Graphic
Click

```
void main()
{
        SERVO_TorqCtrl[254]
        motionready( 0 )
        delay( 1500 )
        while( true )
        {
                if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 1 ) )
                {
                        motion( 0 )
                        waitwhile( MPSU_PlayingMotion )
                }
                else
                {
                        if( ( MPSU_ADCType1 == 2 && MPSU_ADCVal1 == 0 ) )
                        {
                                for( i = 1 ~ 2 )
                                {
                                        motion( 1 )
```

06 定数の設定

サーボモーターを自分で動かすことができる状態にする方法です。
Variable TypeでConstantを選択します。
プロパティー中のConstant　Value値を96に設定します。サーボのTorqControlレジスタに96(0x60)という値が入ると、サーボを動かすことができる状態になります。
その値はOutputコネクタを通して後ろのモジュールのトルク値に渡されます。


1 Click
Start
Constant
96
254
Servo RAM
TorqCtrl
2 Select
3 Input
MPSU RAM
Servo RAM
Servo ID
Variable Name
True
False

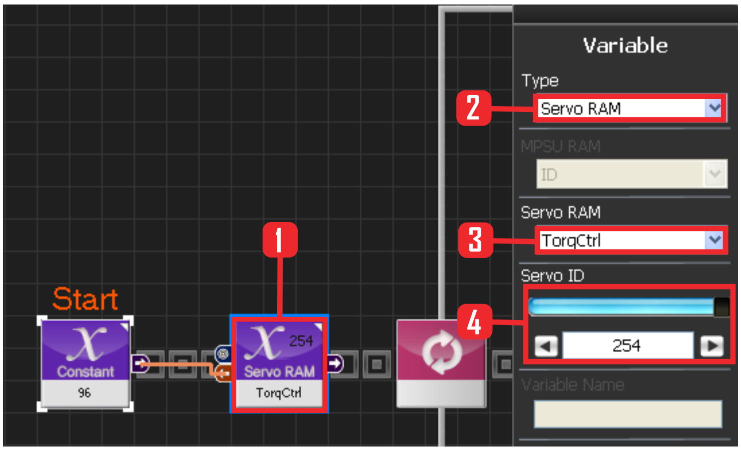
Variable

Type: Constant

Constant Type: ☐ Bool ● Int

Constant Value: 96

07 すべてのサーボに適用

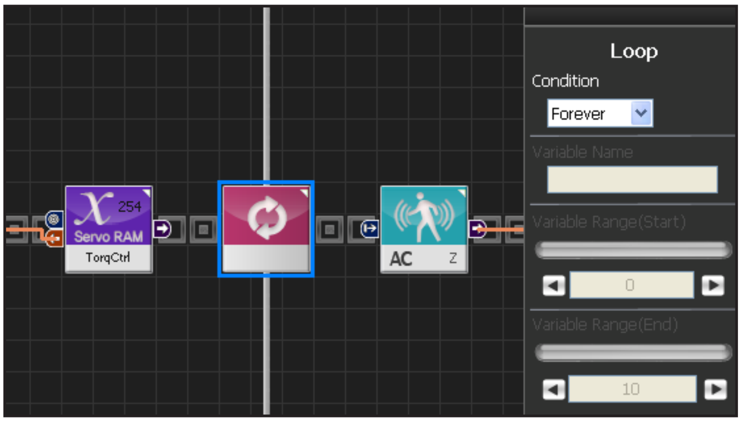

Start
Constant
96
254
Servo RAM
TorqCtrl
1
2
3
4

Variable
Type: Servo RAM
MPSU RAM: ID
Servo RAM: TorqCtrl
Servo ID: 254
Variable Name

前にもらった定数値96をすべてのサーボに適用する方法です。

Variable > Type：Servo RAMを選択します。
Servo RAM：TorqCtrlを選択します。
Servo ID：254を選択します。 254は接続されているすべてのサーボに適用するという意味です。

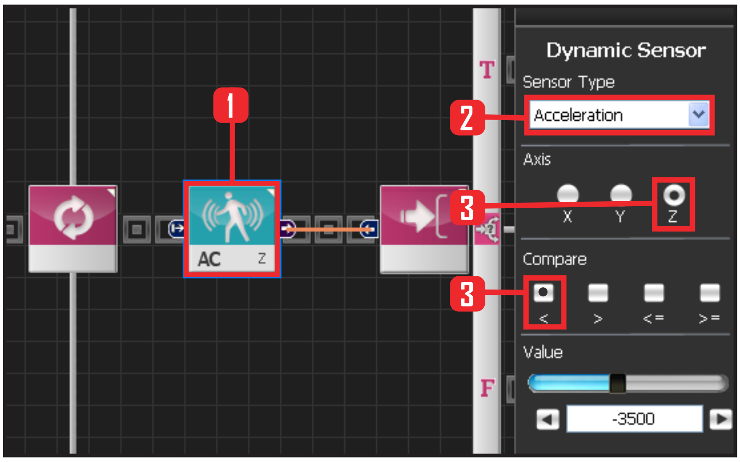
08 反復


254
Servo RAM
TorqCtrl
AC
Z

Loop
Condition: Forever
Variable Name
Variable Range(Start): 0
Variable Range(End): 10

Forever無限反復を設定します。

09 加速度設定（前に倒れた時）

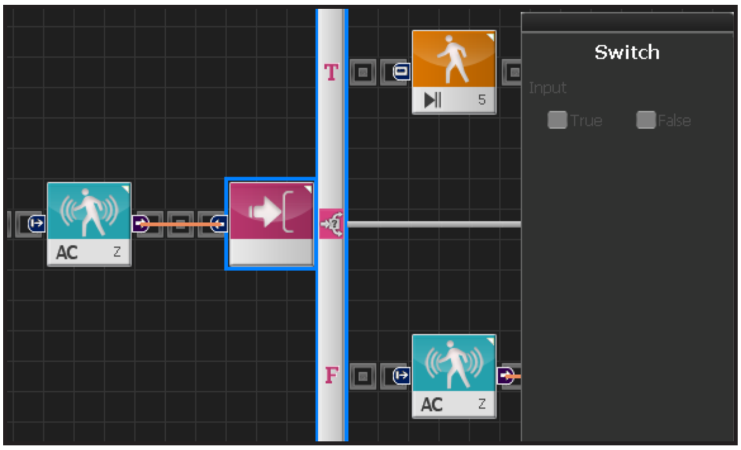


Dynamic Sensor
Sensor Type: Acceleration
Axis: ○ X ○ Y ● Z
Compare: ● < ○ > ○ <= ○ >=
Value: -3500

ロボットが立っているときの加速度は0です。
ロボットが前に倒れたときの加速度は-4096、後ろに倒れたときの加速度は+4096です。
したがって-4096の値に近いときはロボットが前に倒れたと判断します。その基準値を-3500に設定します。

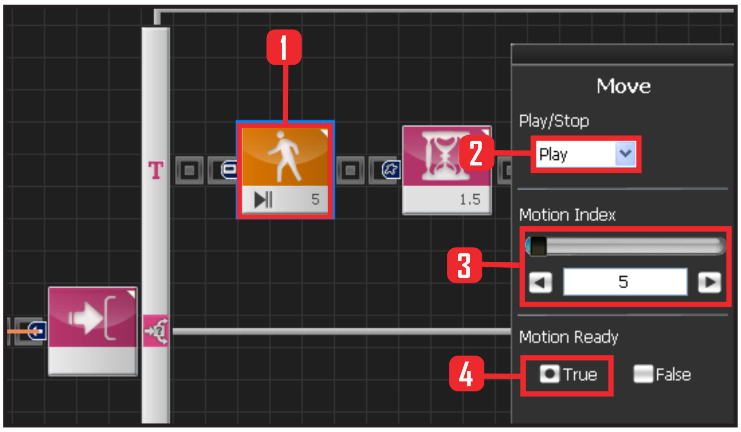
-3500より小さい値の場合、ロボットが前に倒れたと判断します。

Sensor > Dynamic Sensorモジュールを選択します。
Sensor Type：Accelerationを選択します。加速度センサです。
Axis：Z軸に設定します。
Compass：＜に設定します。
Value：-3500に設定します。



10 If 分岐文

Trueの時は後ろに立ち上がるモーションを実行します。Falseの時は次の条件文に移動するプログラミングです。



Move

Play/Stop: Play

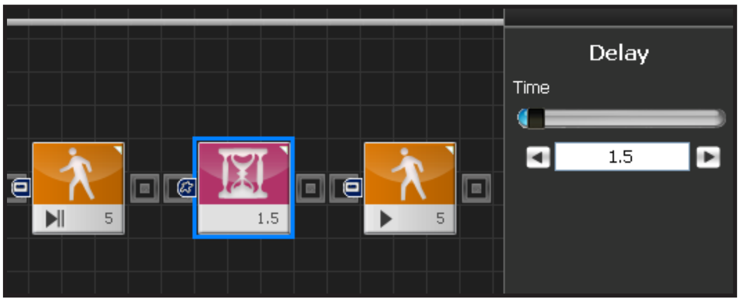
Motion Index: 5

Motion Ready: ☑ True ☐ False

11 後ろに立ち上がるモーション

ロボットが前に倒れたときにはロボットを後ろに立ち上がらせる動作を挿入する必要があります。
5番モーションが後ろに立ち上がるモーションです。

Motion > Moveモジュールを選択します。
Play/ Stop : Playを選択します。
Motion Index : 5番を選択します。 5番は後ろに立ち上がるモーションです。
Motion Ready : Trueを選択します。
モーションの準備過程です。

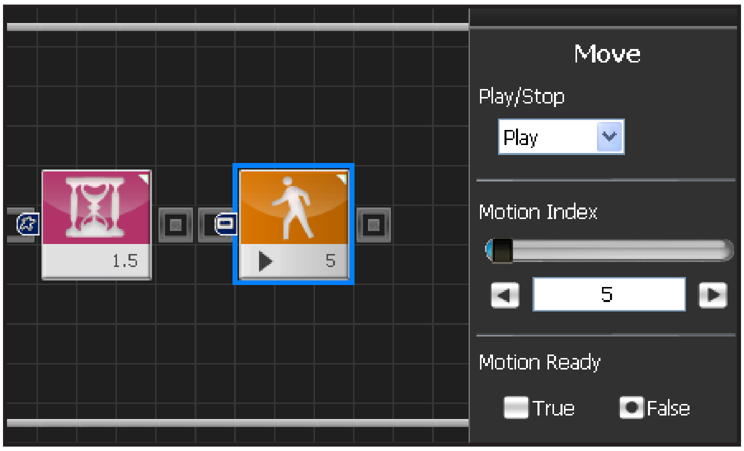


Delay

Time: 1.5

12 Delay

Motion Ready動作が終了する前にモーションが進行することを防止するためにDelay値を1.5秒に設定します。



Move

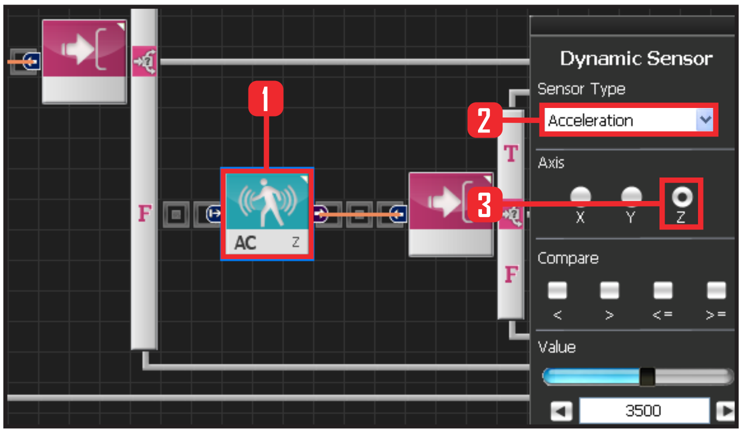
Play/Stop: Play

Motion Index: 5

Motion Ready: ○ True ● False

13 後ろに立ち上がるモーションの実行

Motion　ReadyでFalseを選択すると、後ろに立ち上がるモーションが実行されます。



Dynamic Sensor

Sensor Type: Acceleration

Axis: ○ X ○ Y ● Z

Compare: □ < □ > □ <= □ >=

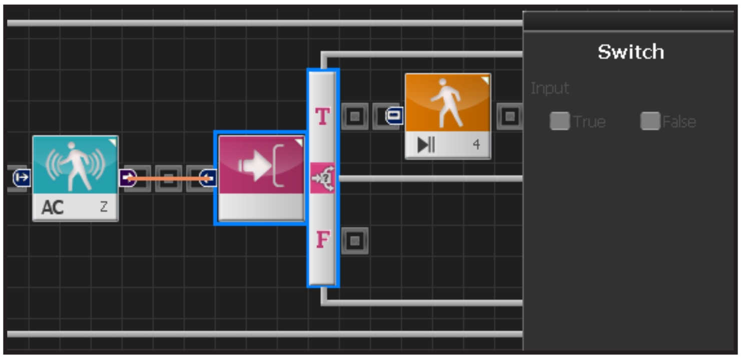
Value: 3500

14 加速度設定（後ろに倒れた時）

ロボットが立っているときの加速度は0です。
ロボットが前に倒れたときの加速度は-4096、後ろに倒れたときの加速度は+4096です。
したがって+4096の値に近いときはロボットが後ろに倒れたと判断します。その基準値を+3500に設定します。

3500より大きい値の場合、ロボットが後ろに倒れたと判断します。

Sensor > Dynamic Sensorモジュールを選択します。
Sensor　Type : Accelerationを選択します。加速度センサです。
Axis : Z軸に設定します。
Compass : <に設定します。
Value : 3500に設定します。

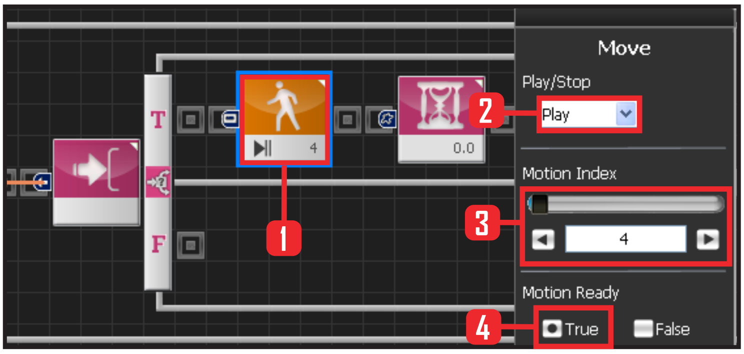


Switch

Input: □ True □ False

15 If 分岐文

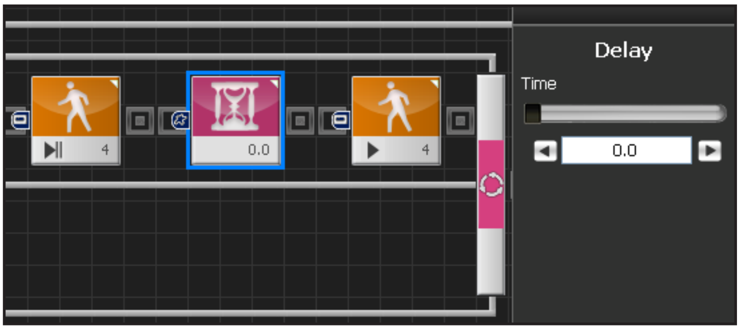
Trueの時は前に立ち上がるモーションを実行するプログラミングです。


Move
Play/Stop: Play
Motion Index: 4
Motion Ready: ● True ☐ False

16 モーション動作の準備

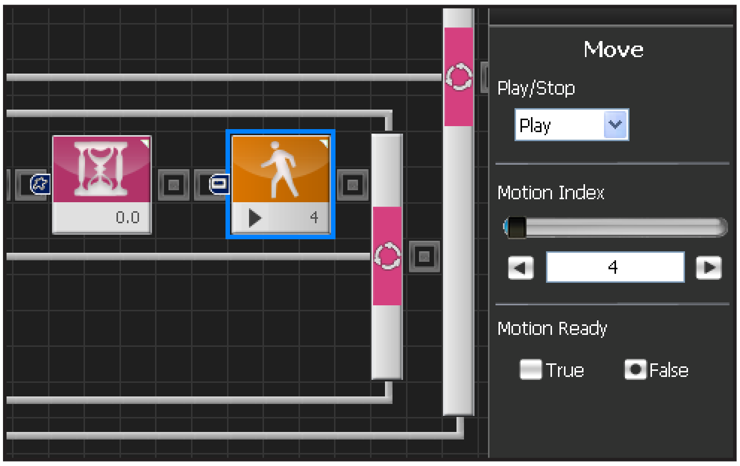
モーション動作のための準備過程です。ロボットのモーションが急に変動した場合、ロボットに無理がかかります。したがって、現在実行しようとするロボットの最初のモーションからゆっくりと移動させます。
Motion ReadyがTrueの場合、モーションの最初のシーンを準備し、Falseの場合、モーションが動作します。

Motion > Moveモジュールを選択します。
Play/ Stop : Playを選択します。
Motion Index : 0番を選択します。 0番は前進しながら歩くモーションです。
Motion Ready : Trueを選択します。
モーションを準備する過程です。


Delay
Time: 0.0

17 Delay

Motion Ready動作が終了する前にモーションが進行することを防止するためにDelay値を1.5秒に設定します。

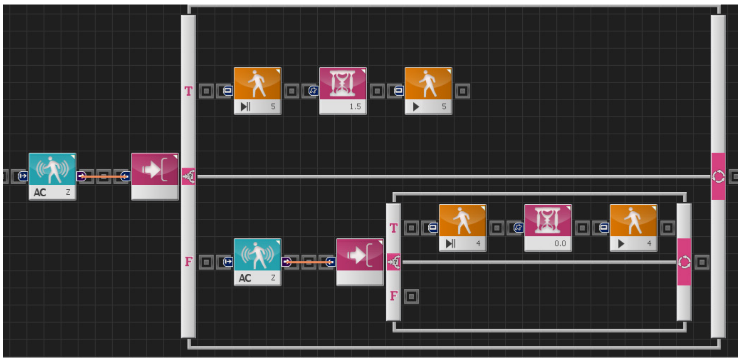

Move
Play/Stop: Play
Motion Index: 4
Motion Ready: ☐ True ● False

18 前に立ち上がるモーションの実行

Motion ReadyをFalseに選択すると、前に立ち上がるモーションが実行されます。

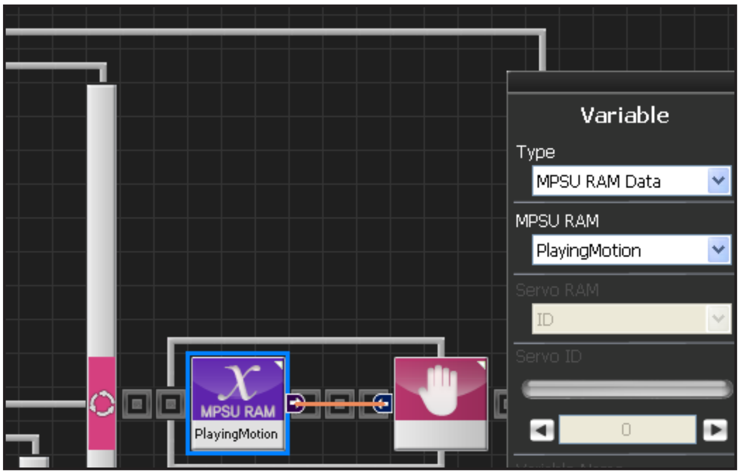
19 倒れたロボットを立ち上がらせる

Z軸の加速度値に基づいてロボットの転倒を判断し、倒れた場合に応じて、それぞれの動作を実行するプログラムです。



20 モーション動作の確認

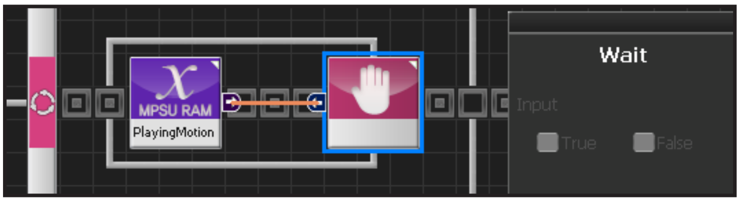
Loopは継続的な繰り返しを意味します。Move命令を下してから、実際のモーションが実行され、完了するまでは時間がかかるので、loop内にMoveモジュールの一つだけを入れて実行すると、モーションが既に実行中央にもかかわらずloopを続けてモーションを実行する命令を繰り返すことになります。
この場合、Moveモジュールを実行した回数と実際のモーションを実行する回数が異なります。
したがって実行したモーションが終わるまで待ってから再びloopの最初に戻す方がより確実です。
Variable > MPSU RAM Dataに入ると、Playing Motion があります。Playing Motionはロボットのモーションを実行しているかを確認する変数です。そのPlaying Motionに Waitをかければ、ロボットの動作が終わるまでLoopを待ちます。
Data > Variableモジュールを選択します。
Type : MPSU RAM Dataを選択します。
MPSU RAM : Playing Motionを選択します。
Outputコネクタの値の後ろのWaitモジュールに接続します。
モーションが終わるまで待つことを意味します。



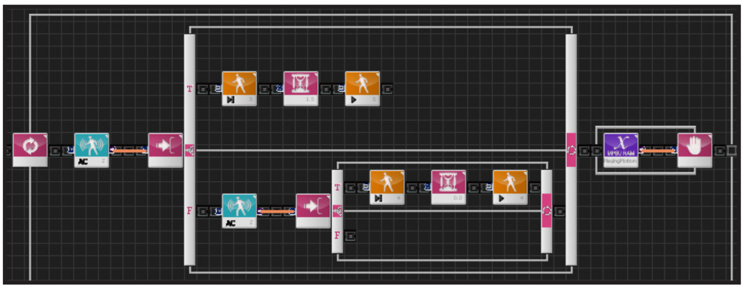
Variable
Type: MPSU RAM Data
MPSU RAM: PlayingMotion
Servo RAM: ID
Servo ID: 0

21 Wait

モーションが終わるまで待ちます。
モーションが終わればまた最初に戻ってモーションを繰り返します。

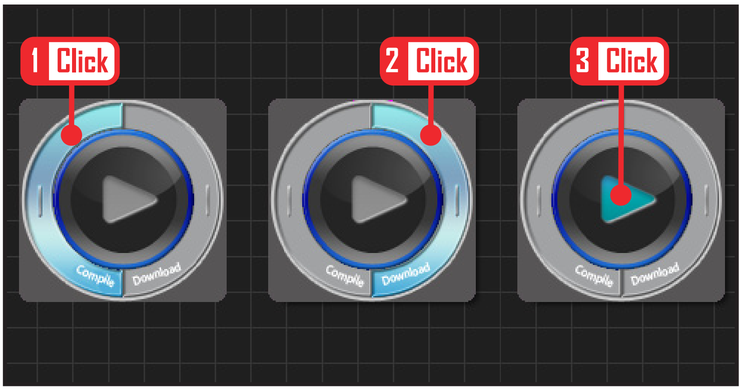


Wait
Input
☐ True ☐ False



22 全体のプログラム

ロボットが前に倒れた時と後ろに倒れた時を判断して、それぞれ後ろに立ち上がるモーション、前に立ち上がるモーションを実行するプログラムです。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

23 コンパイル、ダウンロード、実行

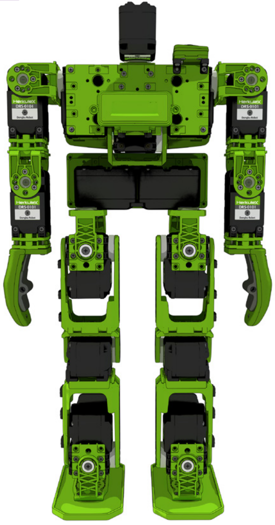
左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。

24 ロボット動作

ロボットが前に倒れている場合は後ろに立ち上がり、ロボットが後ろに倒れている場合は前に立ち上がります。

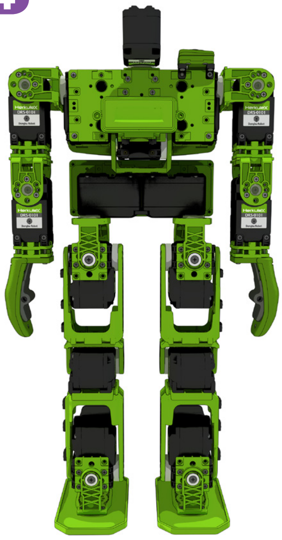
1

2



3

4

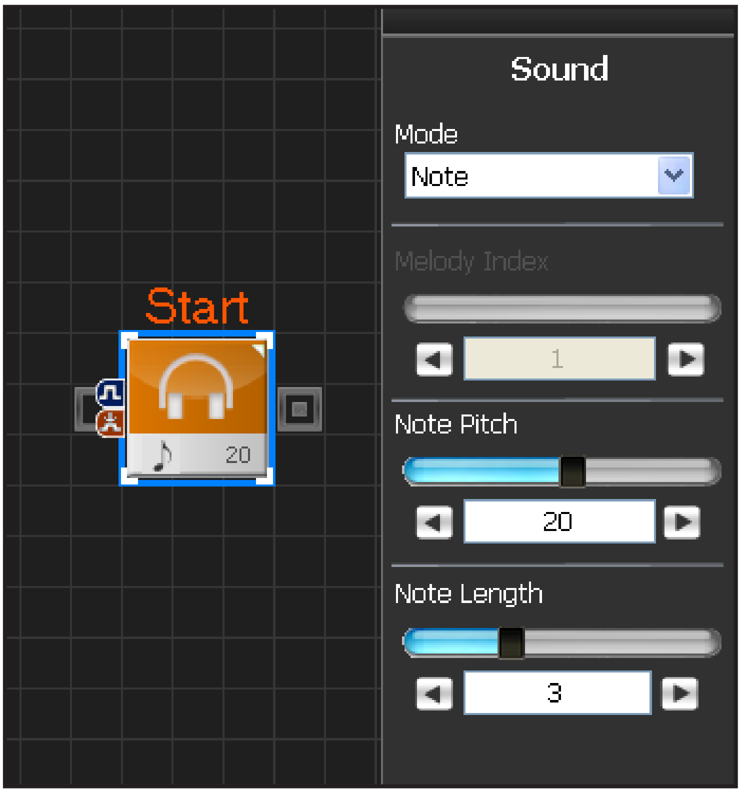


IRReceive、Soundの例

例の説明

リモコン番号とSound音をマッチングさせ、ドレミファソラシド（1番～8番）音を出すプログラムです。

ドレミ音はMotion > Soundモジュールで、NoteのPitch値によって音が異なります。DRCコントローラーのピッチは0番から37番まであり、合計3オクターブまで音を出すことができます。



Sound

Mode: Note

Melody Index: 1

Note Pitch: 20

Note Length: 3

00 Soundのプロパティーウィンドウ

Motion > Soundモジュールを選択します。
ModeにはMelodyとNoteがあります。 Melodyは保存されている編集した音リストの一つを選択して再生します。
ModeのNoteは36個の音を取得して使用するときを選択します。
下記の表を参照してください

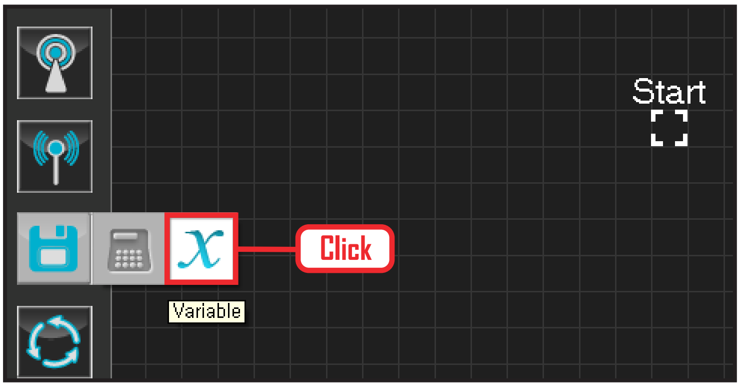
Note Pitchは0番から37番までを選択します。3オクターブで構成されています。
Note Lengthは音のテンポを意味します。32分音符から全音符まで選択することができます。
下記の表を参照してください。

Note Pitch (音の高さ)

番号	0												
階名	無音												
番号	1	2	3	4	5	6	7	8	9	10	11	12	
階名	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ	
番号	13	14	15	16	17	18	19	20	21	22	23	24	
階名	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ	
番号	25	26	27	28	29	30	31	32	33	34	35	36	37
階名	ド	ド#	レ	レ#	ミ	ファ	ファ#	ソ	ソ#	ラ	ラ#	シ	ド

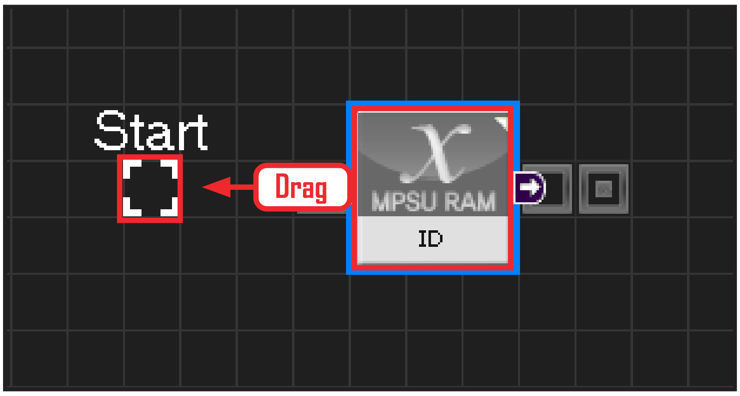
Note Length (音の長さ)

番号	0	1	2	3	4	5	6	7	8	9
Raw Data	6	12	18	24	36	48	72	96	144	192
時間 (ms)	38.4	76.8	115.2	153.6	230.4	307.2	460.8	614.4	921.6	1228.8
音符	32分音符	16分音符	付点16分音符	8分音符	付点8分音符	4分音符	付点4分音符	2分音符	付点2分音符	全音符


Start
Click
Variable

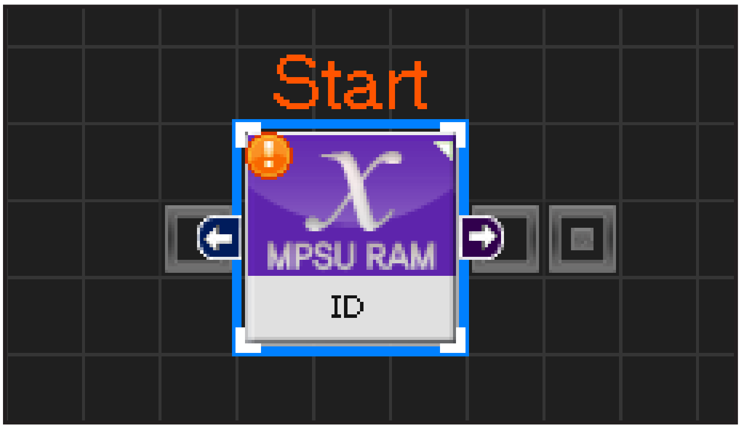
01 変数の指定

Data > Variableモジュールをクリックします。


Start
Drag
MPSU RAM
ID

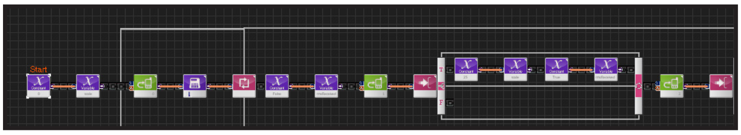
02 開始

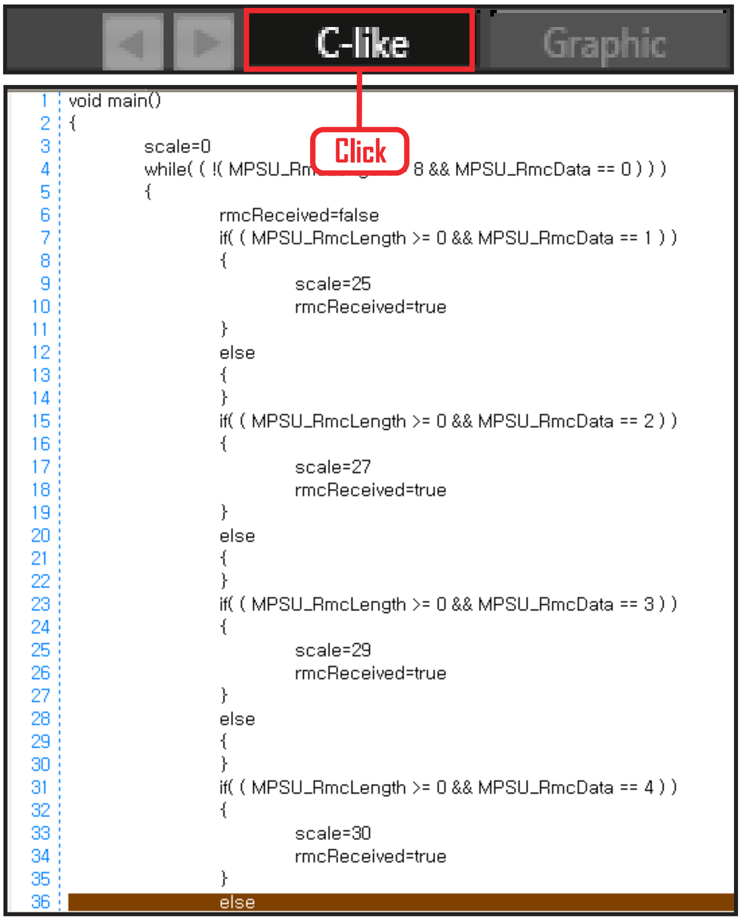
モジュールの左側の連結線をStartPointにドラッグして正確にドッキングさせます。


Start
MPSU RAM
ID

03 プログラミングの開始

モジュールとStart Pointが正確にドッキングすると、左側のように有効化されたカラーイメージのモジュールに変わります。
これはプログラミングが開始されたという意味です。





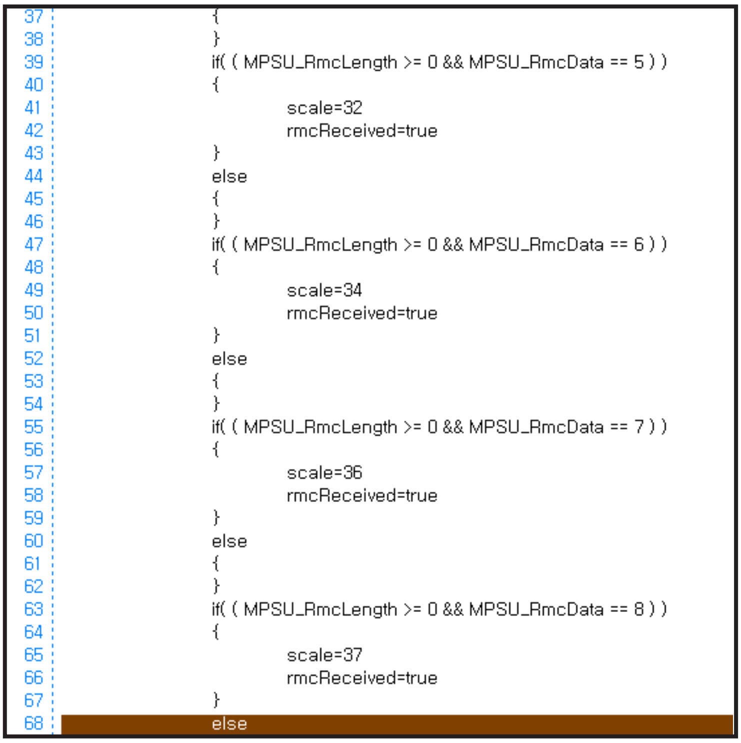
```
void main()
{
        scale=0
        while( ( !( MPSU_Rm            8 && MPSU_RmcData == 0 ) ) )
        {
                rmcReceived=false
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 1 ) )
                {
                        scale=25
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 2 ) )
                {
                        scale=27
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 3 ) )
                {
                        scale=29
                        rmcReceived=true
                }
                else
                {
                }
                if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 4 ) )
                {
                        scale=30
                        rmcReceived=true
                }
                else
```

04 完全プログラミング

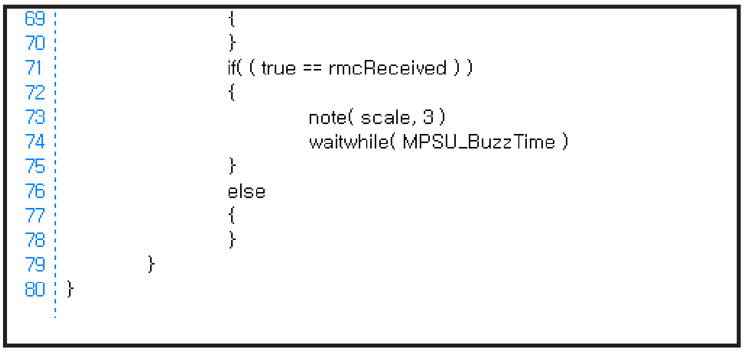
リモコンとブザーを利用した完全なプログラムです。

05 C-Like

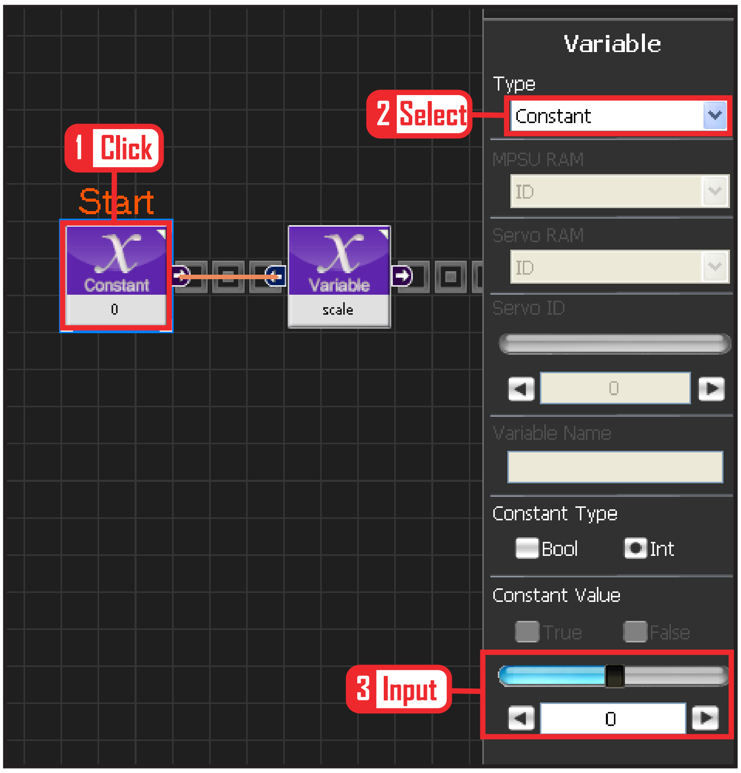
右上のGraphicタブで、C-likeタブをクリックすると、左側のようなTaskプログラミングの画面が表示されます。
IRReceiveリモコンとサウンドを利用して完全なプログラミングをする画面です。
Cと似ている文法構造を持っているのでC文法の先行学習の効果もあります。
各モジュールごとにクリックすると、カーソルが今のプログラムの位置を示しながらモジュールごとにTextにどのように変換するかを確認することができます。



```
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 5 ) )
        {
            scale=32
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 6 ) )
        {
            scale=34
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 7 ) )
        {
            scale=36
            rmcReceived=true
        }
        else
        {
        }
        if( ( MPSU_RmcLength >= 0 && MPSU_RmcData == 8 ) )
        {
            scale=37
            rmcReceived=true
        }
        else
```



```
        {
        }
        if( ( true == rmcReceived ) )
        {
            note( scale, 3 )
            waitwhile( MPSU_BuzzTime )
        }
        else
        {
        }
    }
}
```


1 Click
Start
Constant
0
Variable
scale
2 Select
3 Input
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
Variable Name
True
False

Type: Constant

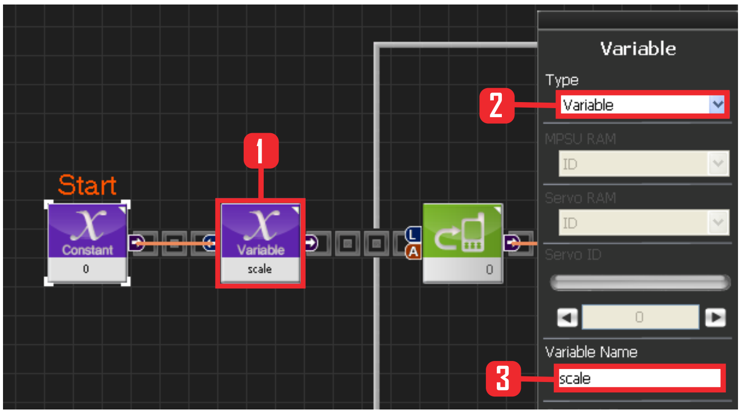
Constant Type: □Bool ●Int

Constant Value: 0

06 定数の設定

プレイ音階の変数を宣言する過程です。

Data > Variableを選択します。
Type : Constantを選択します。
Constant Value : 0に設定します。


Start
Constant
0
1
Variable
scale
L
A
0
2
3
MPSU RAM
ID
Servo RAM
ID
Servo ID
0

Variable

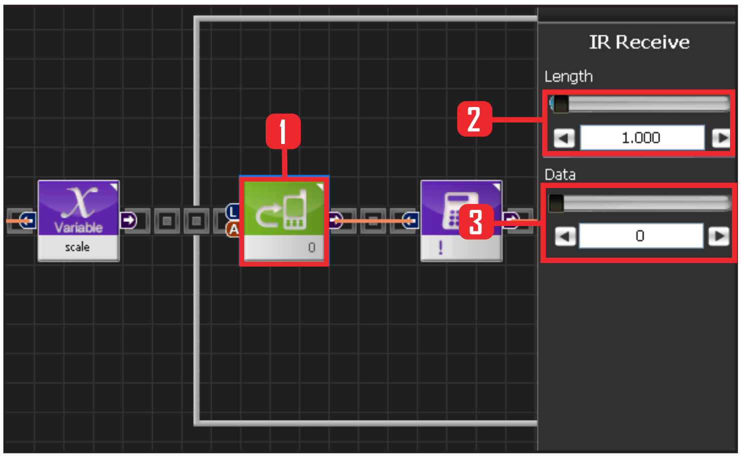
Type: Variable

Variable Name: scale

07 変数名

Playする音階の変数名を指定します。

Data> Variableを選択します。
Type : Variableを選択します。
Variable Name : scaleで指定します。


Variable
scale
1
L
A
0
2
3
!

IR Receive

Length: 1.000

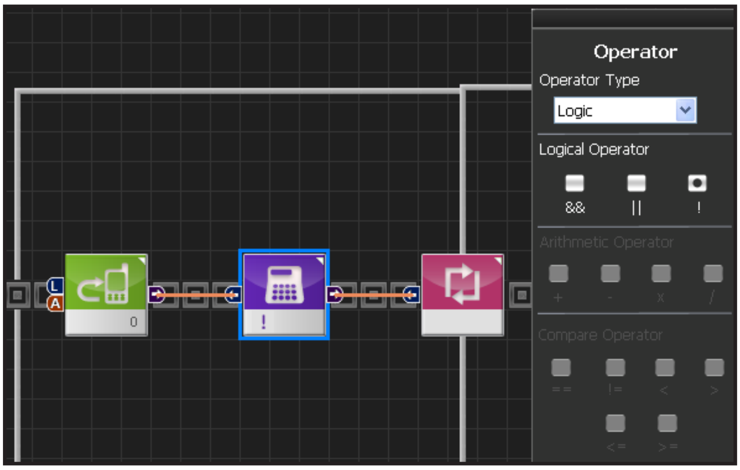
Data: 0

08 While 文の例外条件

リモコンの0番を長く押すと、終了します。

Communication>IRReceiveモジュールを選択します。
Length : 1.000に設定します。1秒間押すことを意味します。
Data : 0に設定します。電源ボタンを押すことを意味します。

電源ボタンを1秒以上長く押すと、このモジュールのOutput値はTrueになり、その他の場合はFalseになります。


L
A
0
!
&&
||
!
Arithmetic Operator
+
-
x
/
Compare Operator
==
!=
<
>
<=
>=

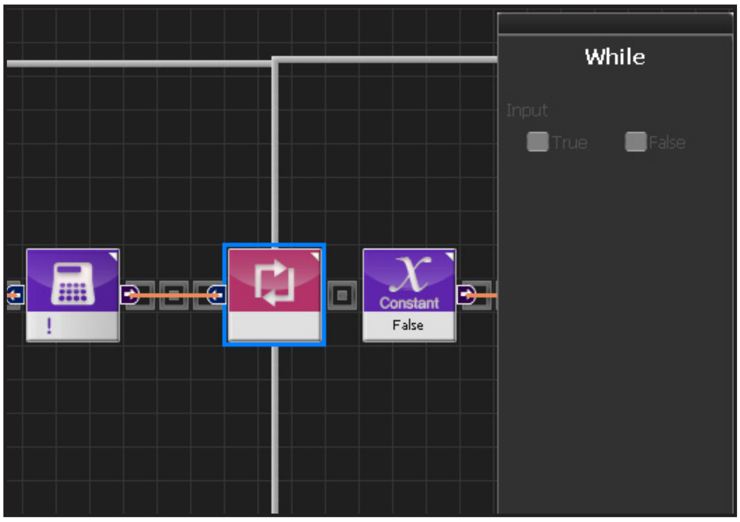
Operator

Operator Type: Logic

Logical Operator: !

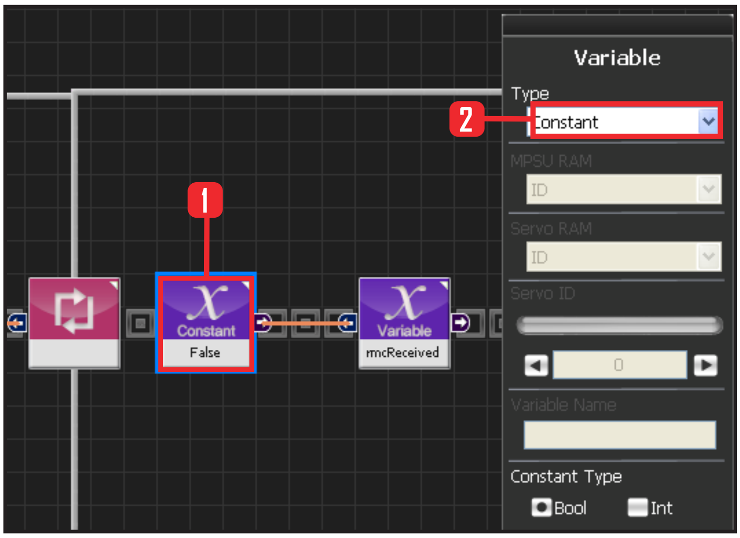
09 !演算子の設定

！演算子はTrue/Falseの値を逆に変更します。したがってIRReceiveモジュールの出力が逆になってwhile文の入力に入ります。


While
Input
True
False
!
Constant
False

10 While 条件の反復

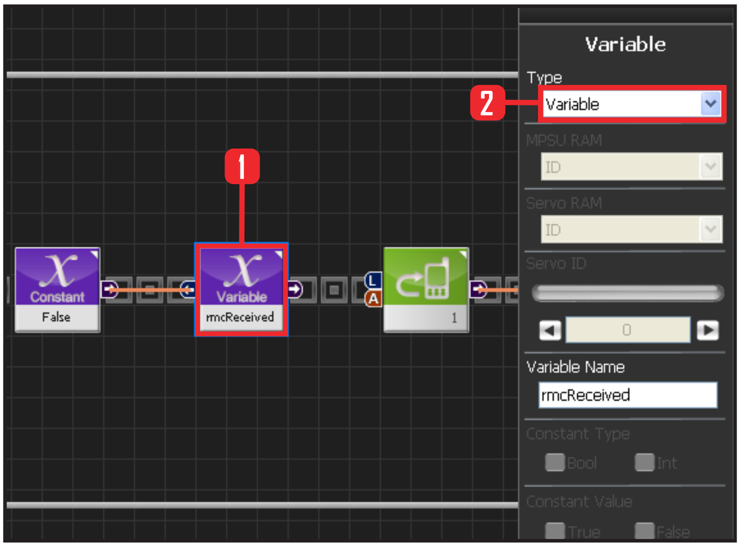
前の条件に応じて繰り返します。
入力がTrueの場合は続けて後の内容を繰り返します。　！演算子によってIRReceiveモジュールのOutputがFalseの場合は繰り返し、Trueの場合はloopを終了します。つまり、電源ボタンを1秒以上長く押すと、ループを抜け出します。


Variable
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
Bool
Int
1
2
Constant
False
Variable
rmcReceived

11 リモコンの入力変数の初期化

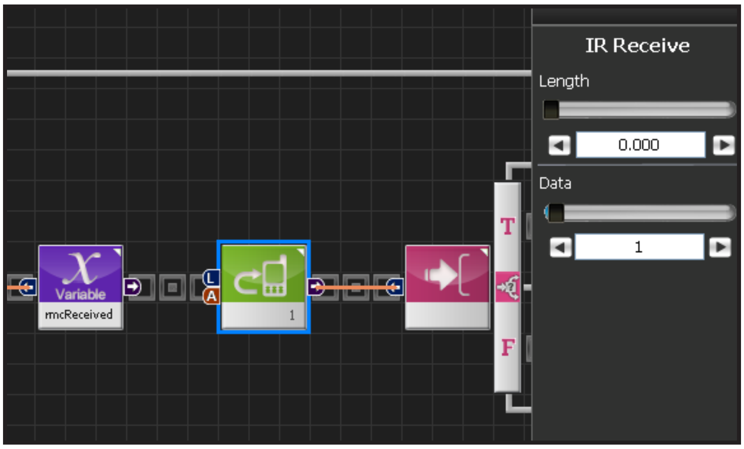
リモコンを入力の変数を設定します。

Data > Variableモジュールを選択します。
Type：Contantを選択します。
Constant Typeは、Boolに設定します。BoolはTrueとFalseを表すデータ型です。
Constant Value：Falseを選択します。


Variable
Type
Variable
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
rmcReceived
Constant Type
Bool
Int
Constant Value
True
False
1
2
Constant
False
Variable
rmcReceived
L
A
1

12 リモコンの入力の初期変数

Data > Variableを選択します。
Type：Variableを選択します。
Variable Name：rmcReceivedを入力する。
rmcReceivedはloopで1～8のリモコンの入力を受け取ったことを示す変数名です。loop（繰り返し）の最初はFalseに初期化して、loop（繰り返し）が終わる時に確認してTrueの場合は音符が再生されます。

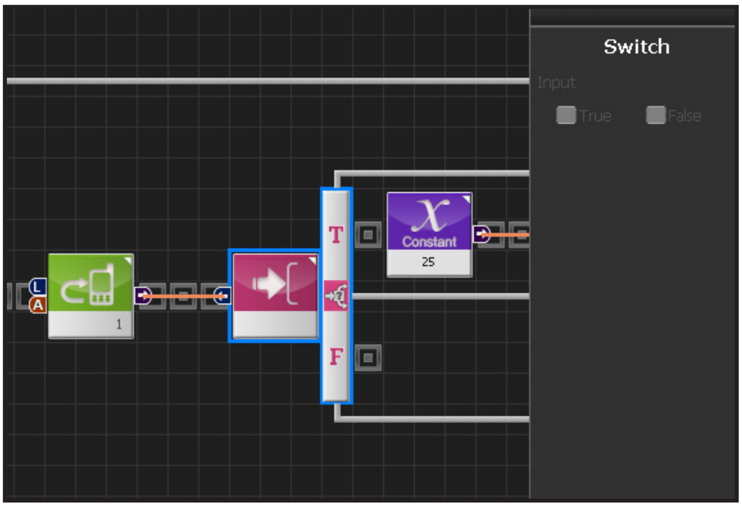

Variable
rmcReceived
L
A
1
T
F

IR Receive
Length
0.000
Data
1

13 1番のリモコン

1番のリモコンが押されているかを確認します。

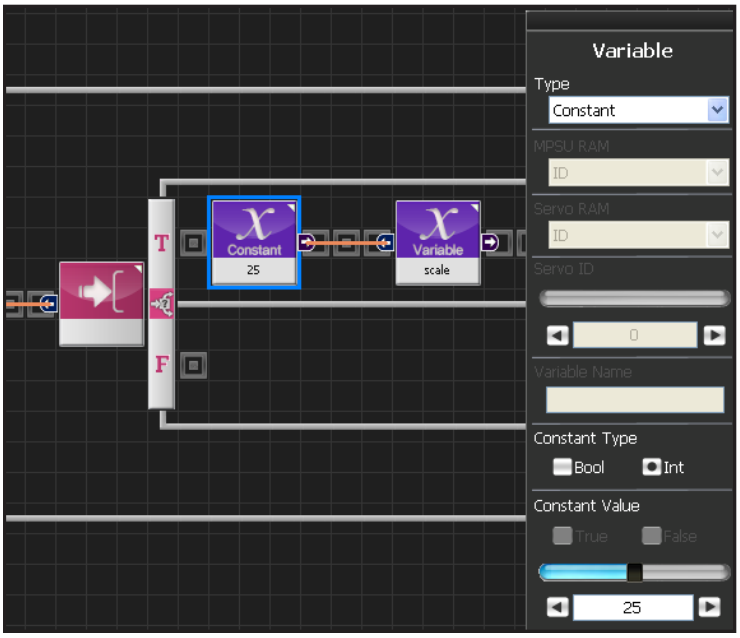
Communication>IRReceiveモジュールを選択します。
Length：0.000に設定します。
Data：1に設定します。1番ボタンを意味します。


L
A
1
T
F
Constant
25

Switch
Input
☐True ☐False

14 IF 条件文

Trueの場合は実行します。


T
F
Constant
25
Variable
scale

Variable
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
☐Bool ●Int
Constant Value
☐True ☐False
25

15 “ド” 音の保存

前に説明したNotePitch（3オクターブ音）の数が25番の場合、”ド”音を意味します。
Scaleの値を”ド”に変換します。

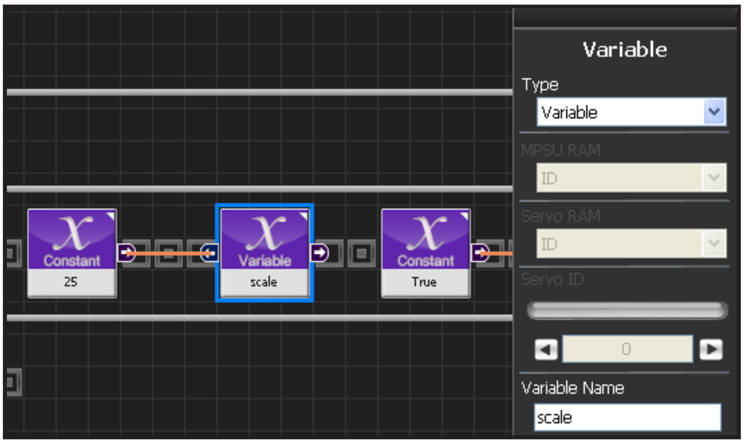
Data > Variableモジュールを選択します。
Type：Contantを選択します。
Constant Typeはintに設定します。
Constant Value：25を選択します。 25は音階”ド”を意味します。

16 scale

Playする音階の変数名はscaleと同じです。

Data > Variableを選択します。
Type : Variableを選択します。
Variable Name : scaleを指定します。

前の定数25の値をinputコネクターに入力します。


Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

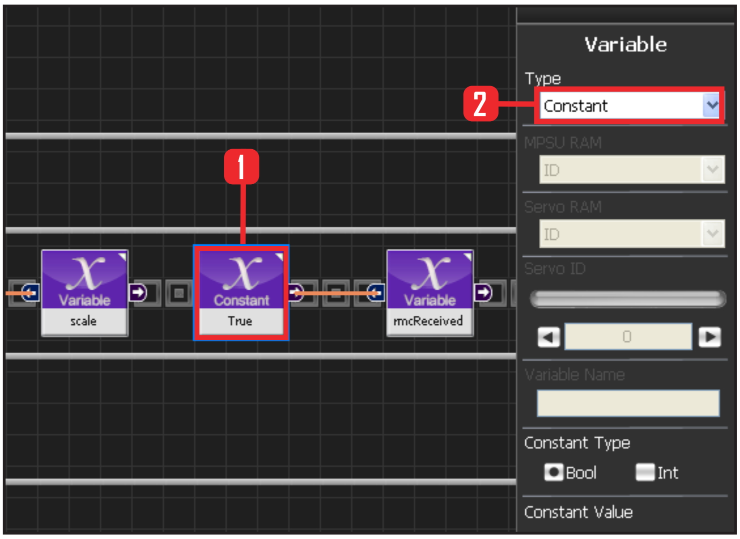
Variable Name: scale

Constant
25
Variable
scale
Constant
True

17 リモコンから信号を受けたことを保存

変数rmcRecievedがTrueの場合、1~8番のボタンが押されたことを意味します。

Data > Variableモジュールを選択します。
Type : Contantを選択します。
Constant Typeは、Boolに設定します。
Constant Value : Trueを選択します。


Variable

Type: Constant

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

Variable Name

Constant Type: ● Bool ☐ Int

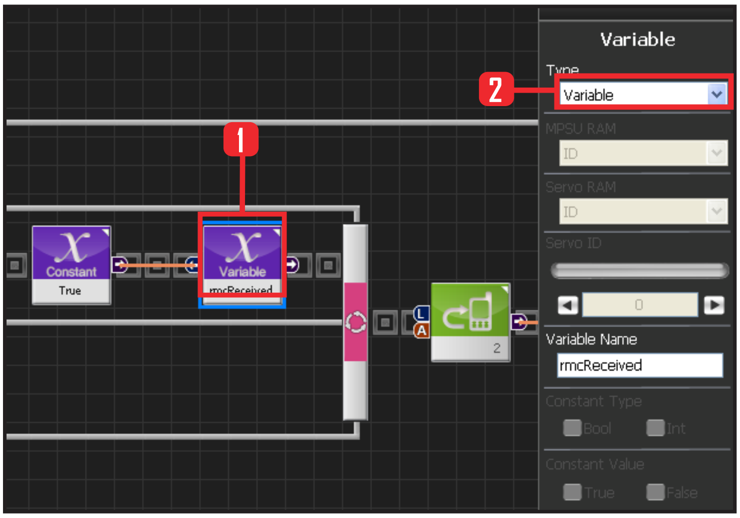
Constant Value

2
1
Variable
scale
Constant
True
Variable
rmcReceived

18 リモコンから信号を受けたことを保存

Data > Variableを選択します。
Type : Variableを選択します。
Variable Name : rmcReceivedと同じです。

前の定数のTrueの値をinputコネクターに入力します。


Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

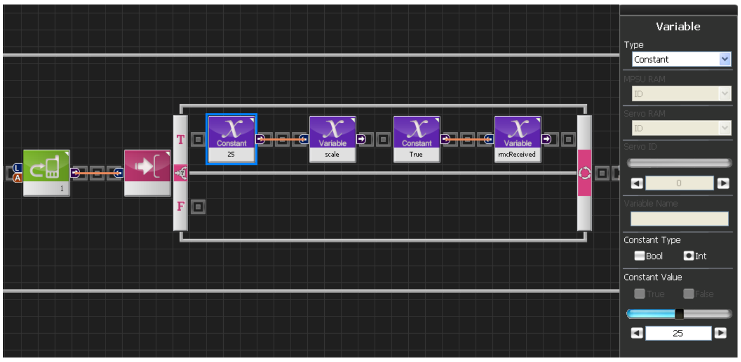
Variable Name: rmcReceived

Constant Type: ☐ Bool ☐ Int

Constant Value: ☐ True ☐ False

2
1
Constant
True
Variable
rmcReceived
L
A
2

19 1番 -> “ド” 音

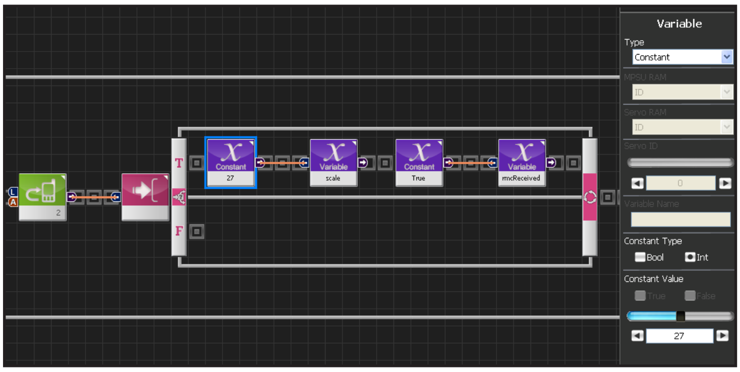

Constant
25
Variable
scale
Constant
True
Variable
rmcReceived
T
F
1

Variable
Type: Constant
Constant Type: ☐Bool ●Int
Constant Value: 25

リモコンの1番を押したときに“ド”音をScaleに保存するプログラムです。

Scale=25番、“ド”音です。

20 2番 -> “レ” 音

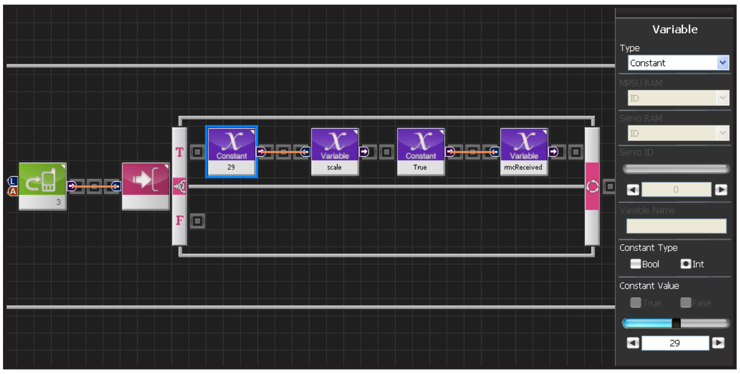

Constant
27
Variable
scale
Constant
True
Variable
rmcReceived
T
F
2

Variable
Type: Constant
Constant Type: ☐Bool ●Int
Constant Value: 27

リモコンの2番を押したときに“レ”音をScaleに保存するプログラムです。

Scale=27番、“レ”音です。

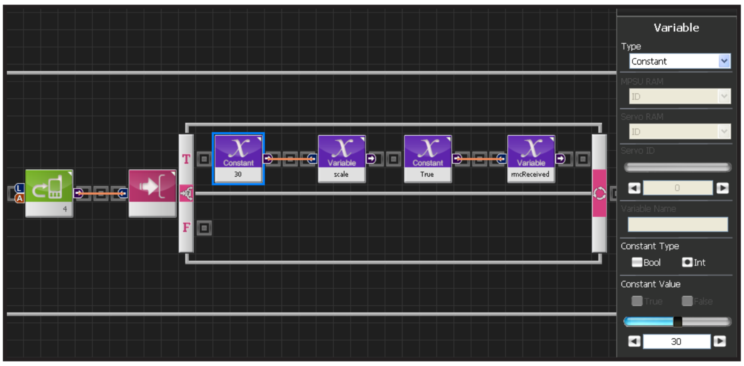
21 3番 -> “ミ” 音


Constant
29
Variable
scale
Constant
True
Variable
rmcReceived
T
F
3

Variable
Type: Constant
Constant Type: ☐Bool ●Int
Constant Value: 29

リモコンの3番を押したときに“ミ”音をScaleに保存するプログラムです。

Scale=29番、“ミ”音です。

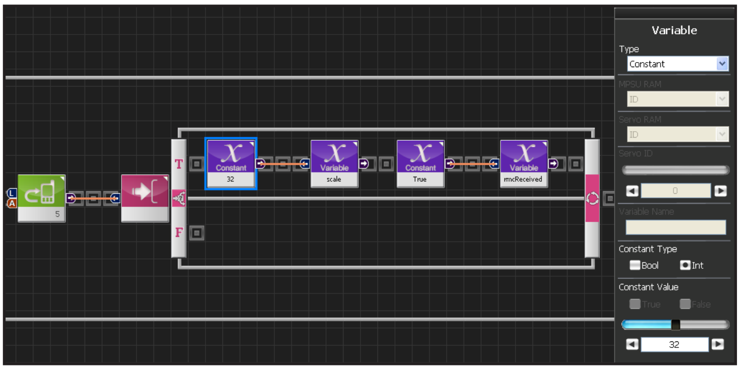

4
T
F
Constant
30
Variable
scale
Constant
True
Variable
rmcReceived
Variable
MPSU RAM
Servo RAM
Servo ID
Variable Name
True
False

Type: Constant
Constant Type: ☐ Bool ● Int
Constant Value: 30

22 4番 -> “ファ” 音

リモコンの4番を押したときに“ファ”音をScaleに保存するプログラムです。

Scale=30番、“ファ”音です。

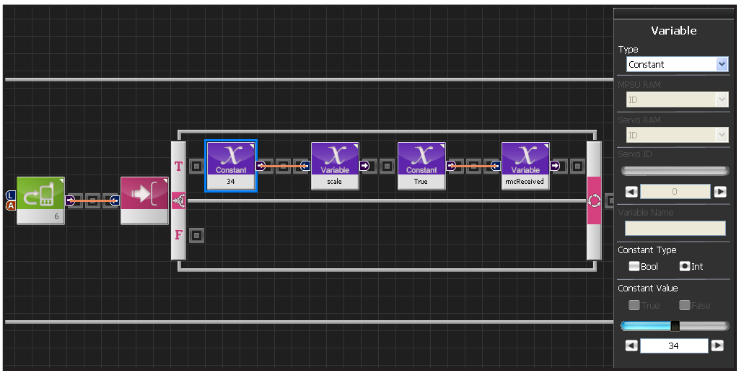

5
T
F
Constant
32
Variable
scale
Constant
True
Variable
rmcReceived
Variable
MPSU RAM
Servo RAM
Servo ID
Variable Name
True
False

Type: Constant
Constant Type: ☐ Bool ● Int
Constant Value: 32

23 5番 -> “ソ” 音

リモコンの5番を押したときに“ソ”音をScaleに保存するプログラムです。

Scale=32番、“ソ”音です。

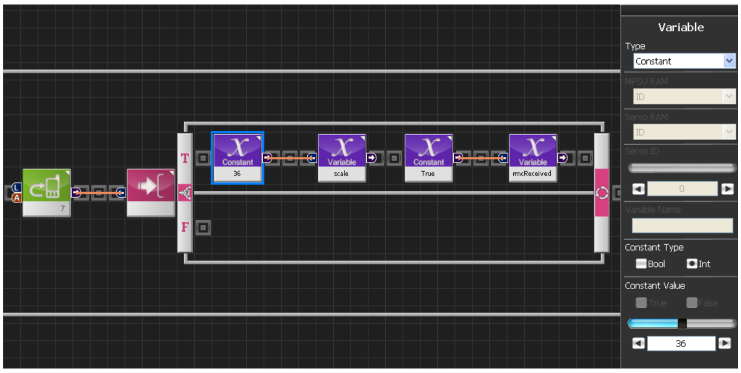

6
T
F
Constant
34
Variable
scale
Constant
True
Variable
rmcReceived
Variable
MPSU RAM
Servo RAM
Servo ID
Variable Name
True
False

Type: Constant
Constant Type: ☐ Bool ● Int
Constant Value: 34

24 6番 -> “ラ” 音

リモコンの6番を押したときに“ラ”音をScaleに保存するプログラムです。

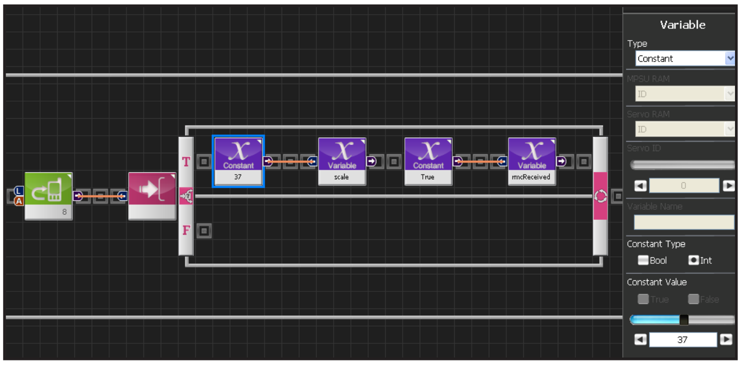
Scale=34番、“ラ”音です。


Variable
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
Bool
Int
Constant Value
True
False
36
T
F
Constant
36
Variable
scale
Constant
True
Variable
rmcReceived
7

25 7番 -> “シ 音

リモコンの7番を押したときに“シ”音をScaleに保存するプログラムです。

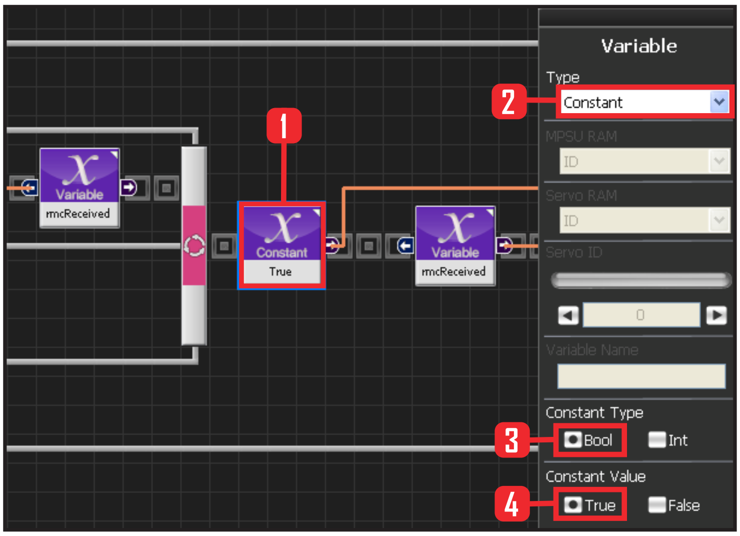
Scale=36番、“シ”音です。


Variable
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
Bool
Int
Constant Value
True
False
37
T
F
Constant
37
Variable
scale
Constant
True
Variable
rmcReceived
8

26 8番 -> “ド” 音

リモコンの8番を押したときに“ド”音をScaleに保存するプログラムです。

Scale=37番、“ド”音です。


Variable
Type
Constant
MPSU RAM
ID
Servo RAM
ID
Servo ID
0
Variable Name
Constant Type
Bool
Int
Constant Value
True
False
1
2
3
4
Variable
rmcReceived
Constant
True
Variable
rmcReceived

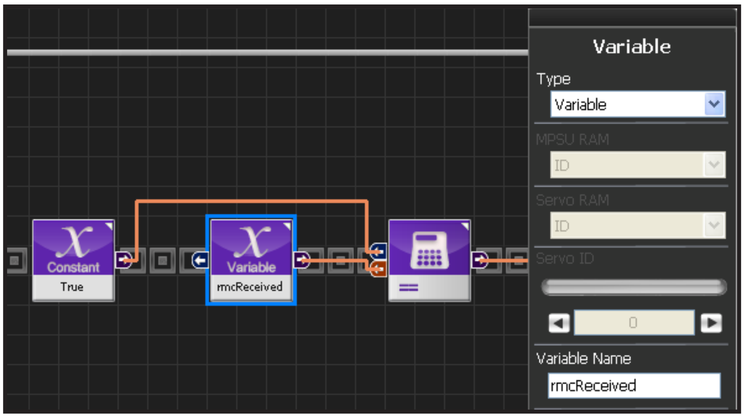
27 rmcReceivedがTrueの場合

rmcReceivedがTrueの場合は前の音の高さの値を保存したscaleの値をnoteに入れて音を再生します。

Data > Variableモジュールを選択します。
Type : Contantを選択します。
Constant Typeは、Boolに設定します。BoolはTrueとFalseを表すデータ型です。
Constant Value : Trueを選択します。

28 rmcReceivedがTrueの場合

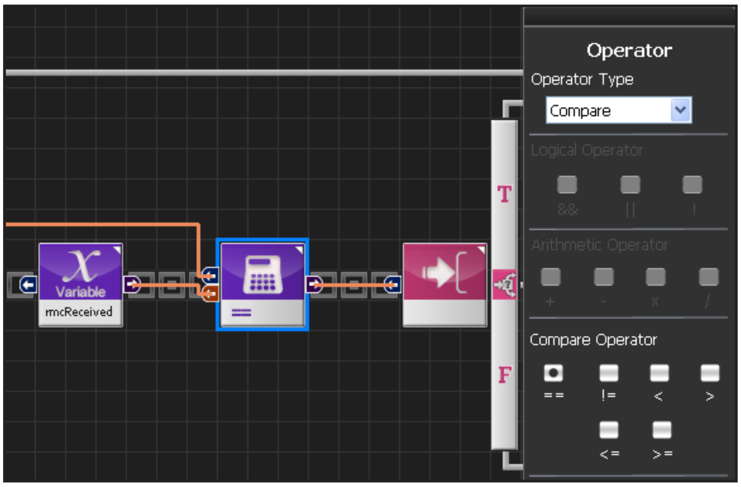
rmcReceived変数名は同じです。


Variable
Type: Variable
MPSU RAM: ID
Servo RAM: ID
Servo ID: 0
Variable Name: rmcReceived
Constant
True
Variable
rmcReceived
==

29 比較演算子 ==

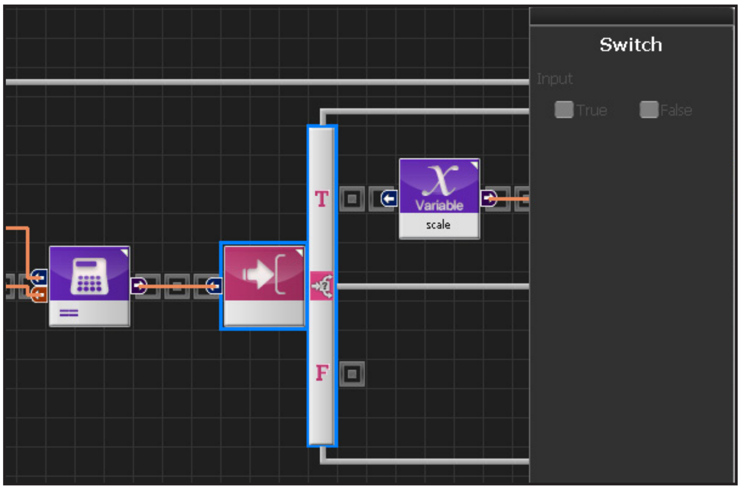
Data > Operatorモジュールを選択します。
Operator Type : Compareを選択します。
Compare Operatorは“==”を選択します。

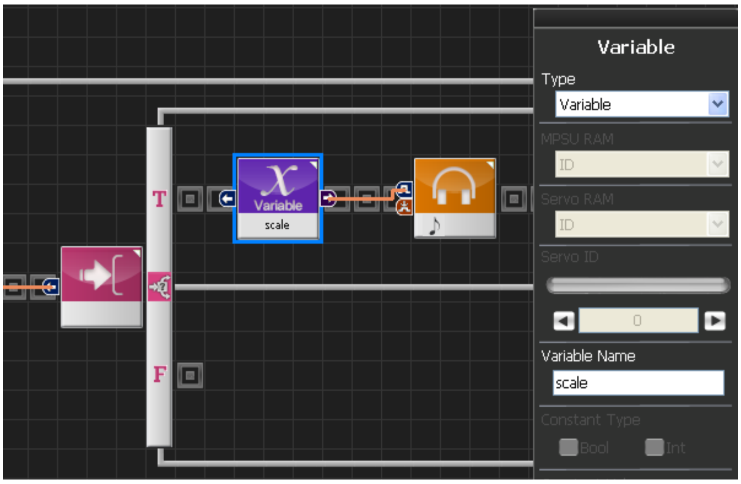
rmcReceived==Trueという意味であり、
”rmcReceivedがTrueと同じだ”ということを示します


Operator
Operator Type: Compare
Logical Operator: ☐ && ☐ || ☐ !
Arithmetic Operator: ☐ + ☐ - ☐ x ☐ /
Compare Operator: ● == ☐ != ☐ < ☐ > ☐ <= ☐ >=
Variable
rmcReceived
==
T
F

30 Switch IF 分岐文

Trueの場合は実行されます。


Switch
Input: ☐ True ☐ False
==
T
F
Variable
scale



Variable

Type: Variable

MPSU RAM: ID

Servo RAM: ID

Servo ID: 0

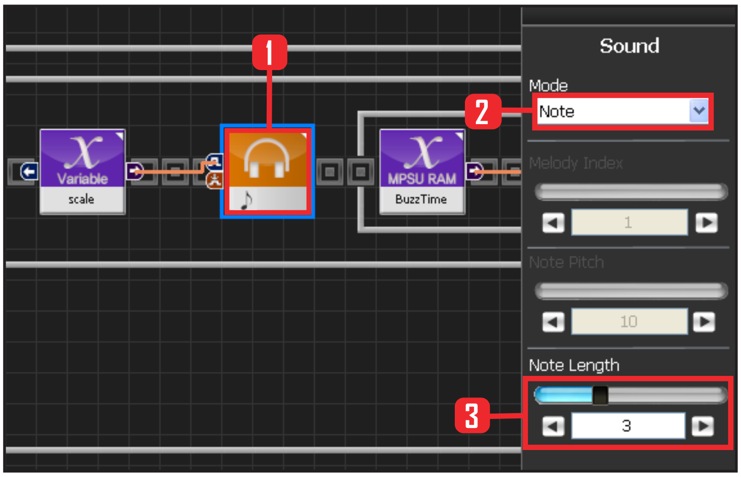
Variable Name: scale

Constant Type: ☐Bool ☐Int

31 scale –> note

Scaleの値をNoteに入れる過程です。

変数scaleモジュールを生成します。



Sound

Mode: Note

Melody Index: 1

Note Pitch: 10

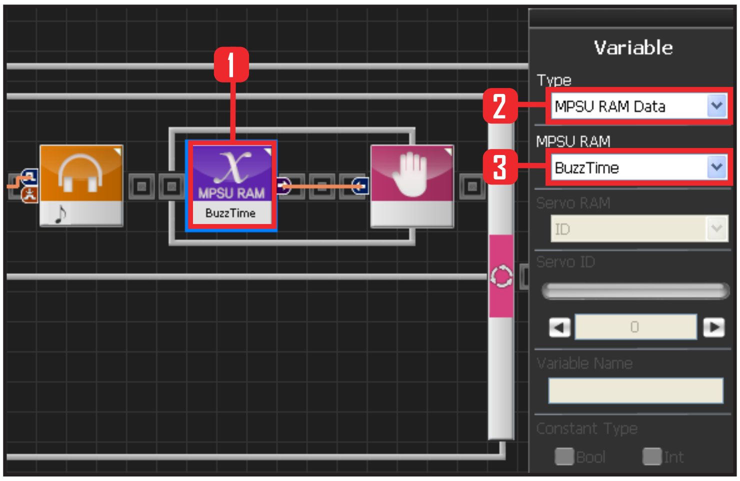
Note Length: 3

32 サウンド Play

Scaleの値をnoteに入れて音が出るようにします。

Motion > Soundモジュールを選択します。
Note Length：3を選択します。
8分音符を意味し、153.6msの間です。

前のリモコンから入力されたボタンに応じてscaleの値を別の方法で保存しており、そのscaleの値をNote Pitchに入れれば、該当する音階が鳴ります。



Variable

Type: MPSU RAM Data

MPSU RAM: BuzzTime

Servo RAM: ID

Servo ID: 0

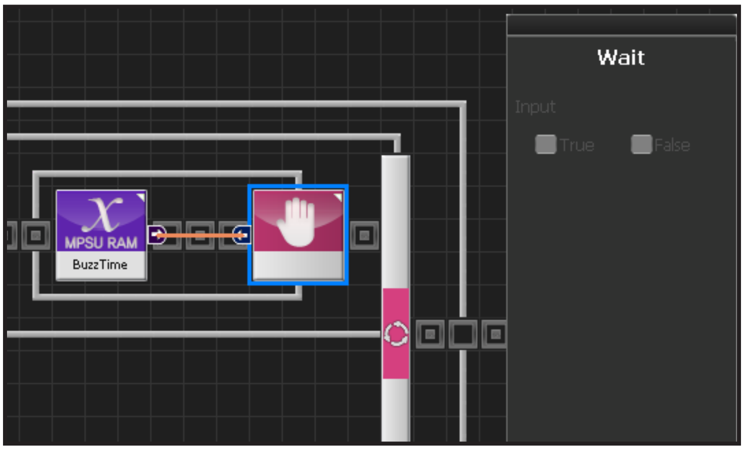
Variable Name:

Constant Type: ☐Bool ☐Int

33 BuzzTime

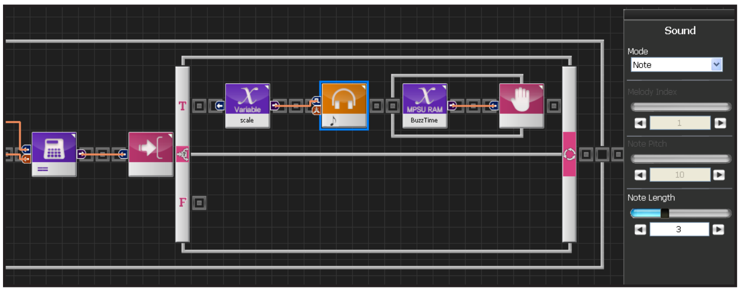
MPSU RAM DataのBuzzTimeは音が出ているかを判断して待ちます。
BuzzTimeはブザーが鳴り始める時、特定の値になった後、6.4msごとに1減少します。
0以外の場合はブザーが鳴っている状態、0の場合はブザーが鳴っていない状態です。
音符による初期BuzzTime値は二番目のページの音の長さの表の”Raw Data”を参照してください。

Data > Variableを選択します。
Type：MPSU RAM Dataを選択します。
MPSU RAM：BuzzTimeを選択します。


Wait
Input
True
False
MPSU RAM
BuzzTime

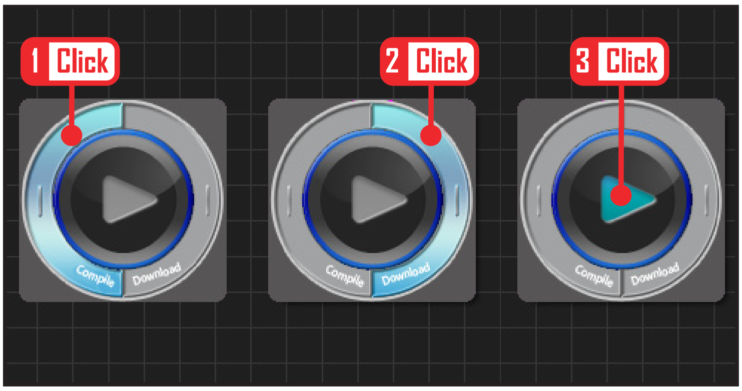
34 Wait

Buzztimeが0になるまで、つまり、音が終了するまで待ちます。


Sound
Mode
Note
Melody Index
1
Note Pitch
10
Note Length
3
T
F
Variable
scale
MPSU RAM
BuzzTime

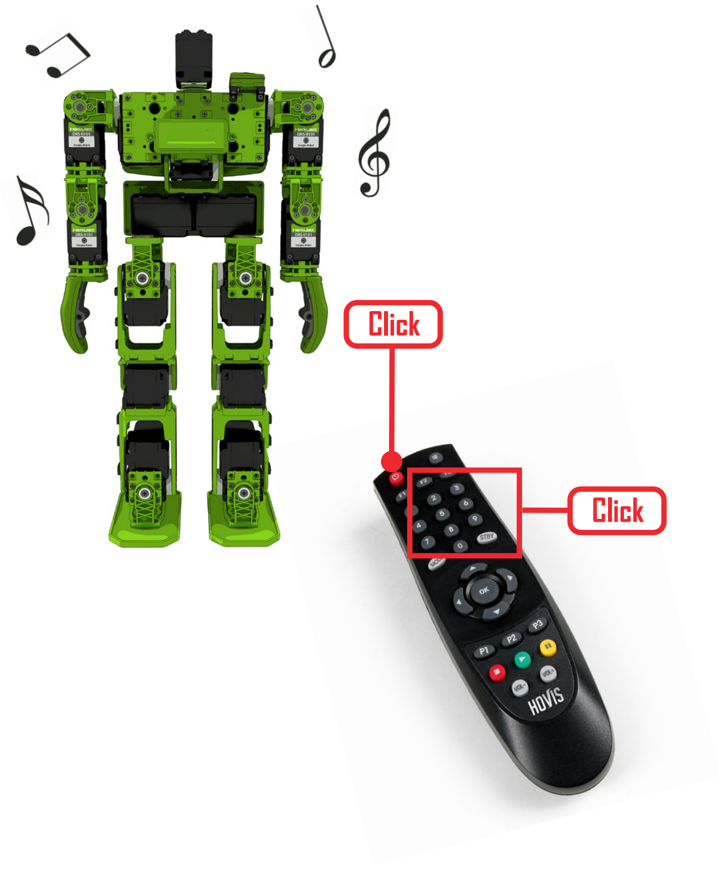
35 音が出る過程

rmcReceivedがTrueのとき、scaleに保存された値をSoundモジュールに入力して、Soundモジュールで音が再生されます。
その音はBuzzTime終了を確認して、もう一度最初に戻ります。


1 Click
2 Click
3 Click
Compile
Download
Compile
Download
Compile
Download

36 コンパイル、ダウンロード、実行

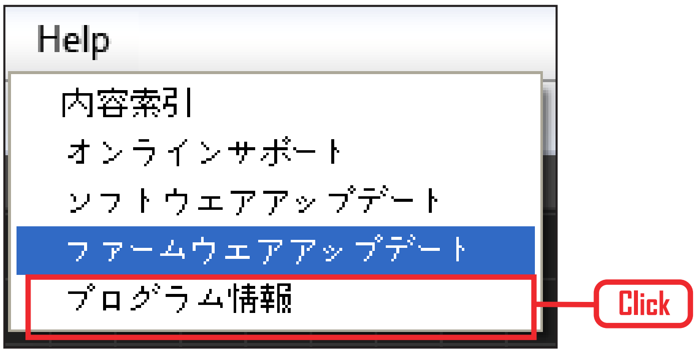
左をクリックしてコンパイルします。エラーがなければ右クリックして、ロボットにダウンロードします。ダウンロードが完了すると、中央の矢印の実行ボタンを押して、ロボットで実行させます。


Click
Click
HOVIS

37 ロボット動作

リモコンの1～8ボタンを押すと、指定した音が出ます。電源ボタンを長く押すとTaskが終了します。

ファームウェアのアップデート


Help
内容索引
オンラインサポート
ソフトウェアアップデート
ファームウェアアップデート
プログラム情報
Click

01　ヘルプ > ファームウェアのアップデート

コントローラーのファームウェアのアップデートはDR-Visual Logicで行います。
コントローラーとPCを接続した状態で、ヘルプ>ファームウェアのアップデートをクリックします。

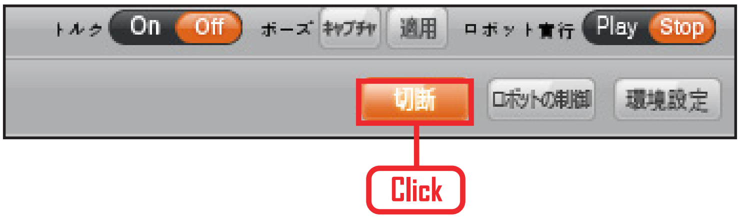
キャリブレーション（ロボットの原点）

ロボットを組み立てた後、正確に組み立てているかを確認して修正する作業です。もし、ロボットが正確に組み立てられていない状態で使用すると誤作動や望ましくない動作をする場合があります。

DR-SIMのロボットの制御をクリックして、実際のロボットモーターの正確な位置を変えながら調整します。

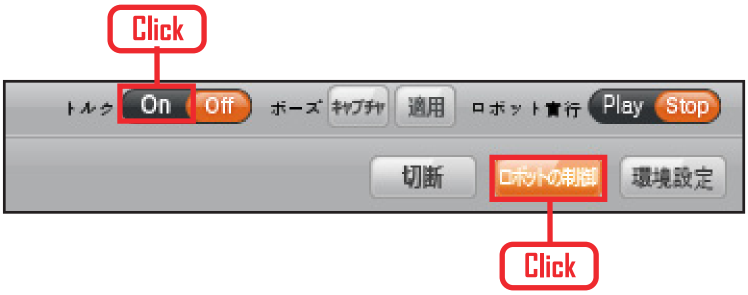
01　接続する

ロボットを接続します。
接続をクリックします。


トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの制御
環境設定
Click

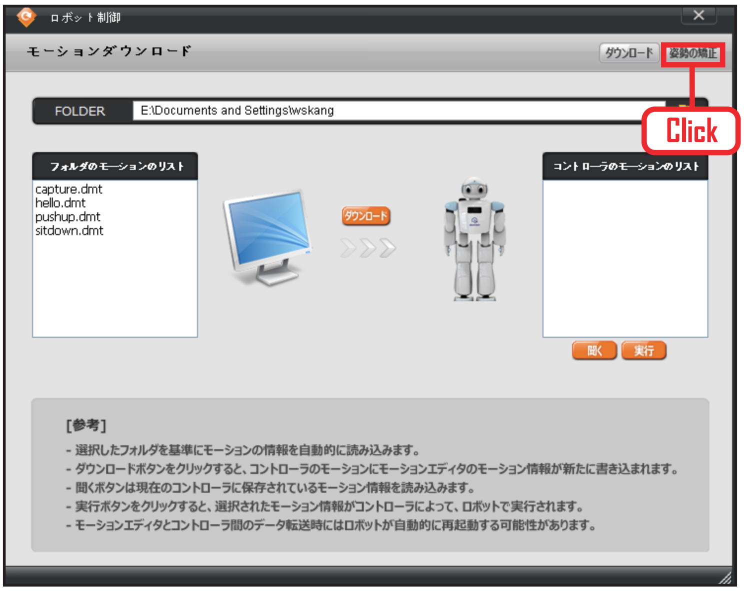
02　ロボット制御

ロボットの電源をオンにします。
トルクONをクリックします。
キャリブレーションはロボット制御にあります。
ロボット制御をクリックします。


Click
トルク
On
Off
ポーズ
キャプチャ
適用
ロボット実行
Play
Stop
切断
ロボットの制御
環境設定
Click

03　姿勢矯正

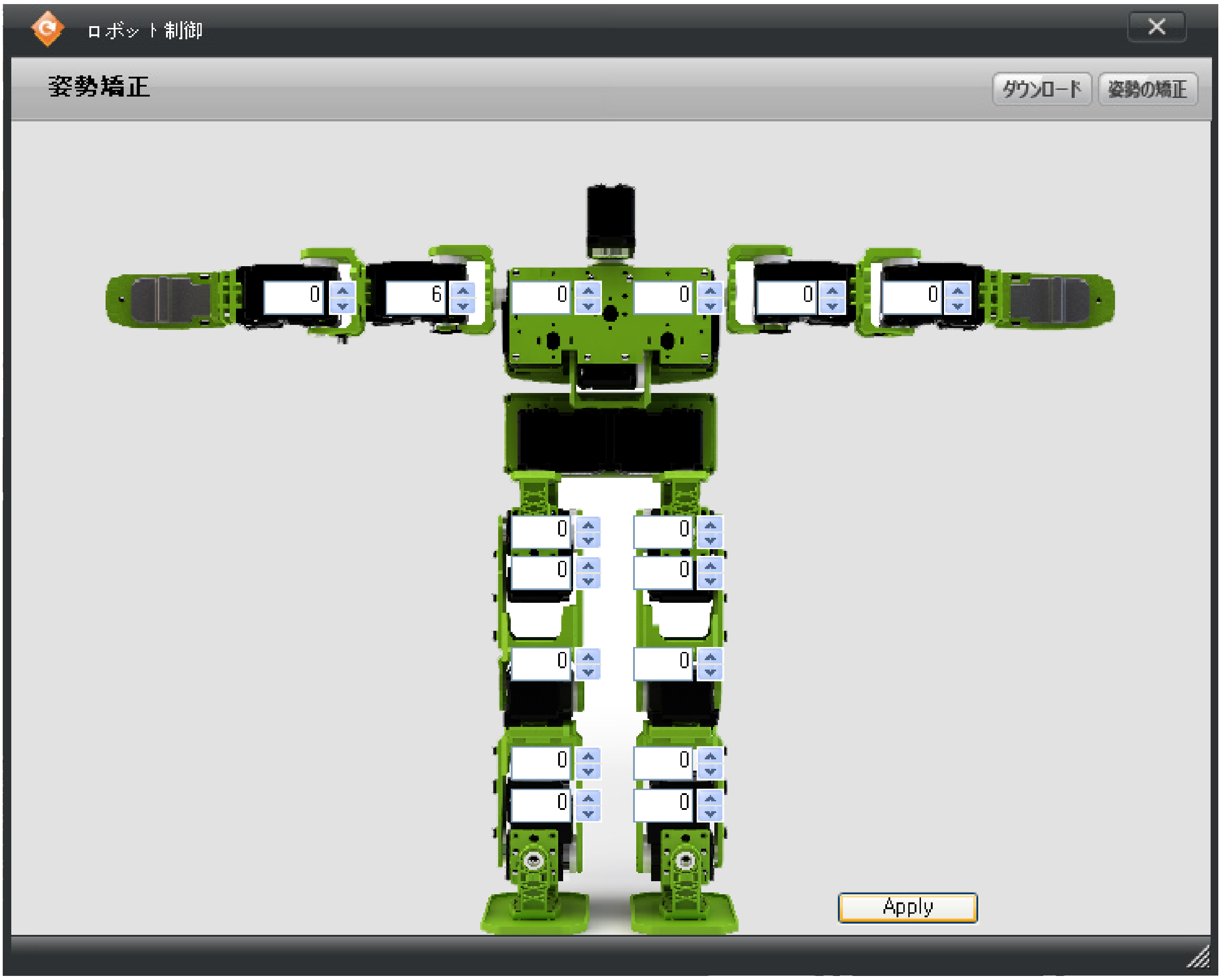
ロボット制御ウィンドウはモーションのダウンロードと姿勢矯正に分かれています。
姿勢矯正をクリックします。


ロボット制御
モーションダウンロード
ダウンロード
姿勢の補正
Click
FOLDER
フォルダのモーションのリスト
ダウンロード
コントローラのモーションのリスト
聞く
実行

FOLDER　E:\Documents and Settings\wskang

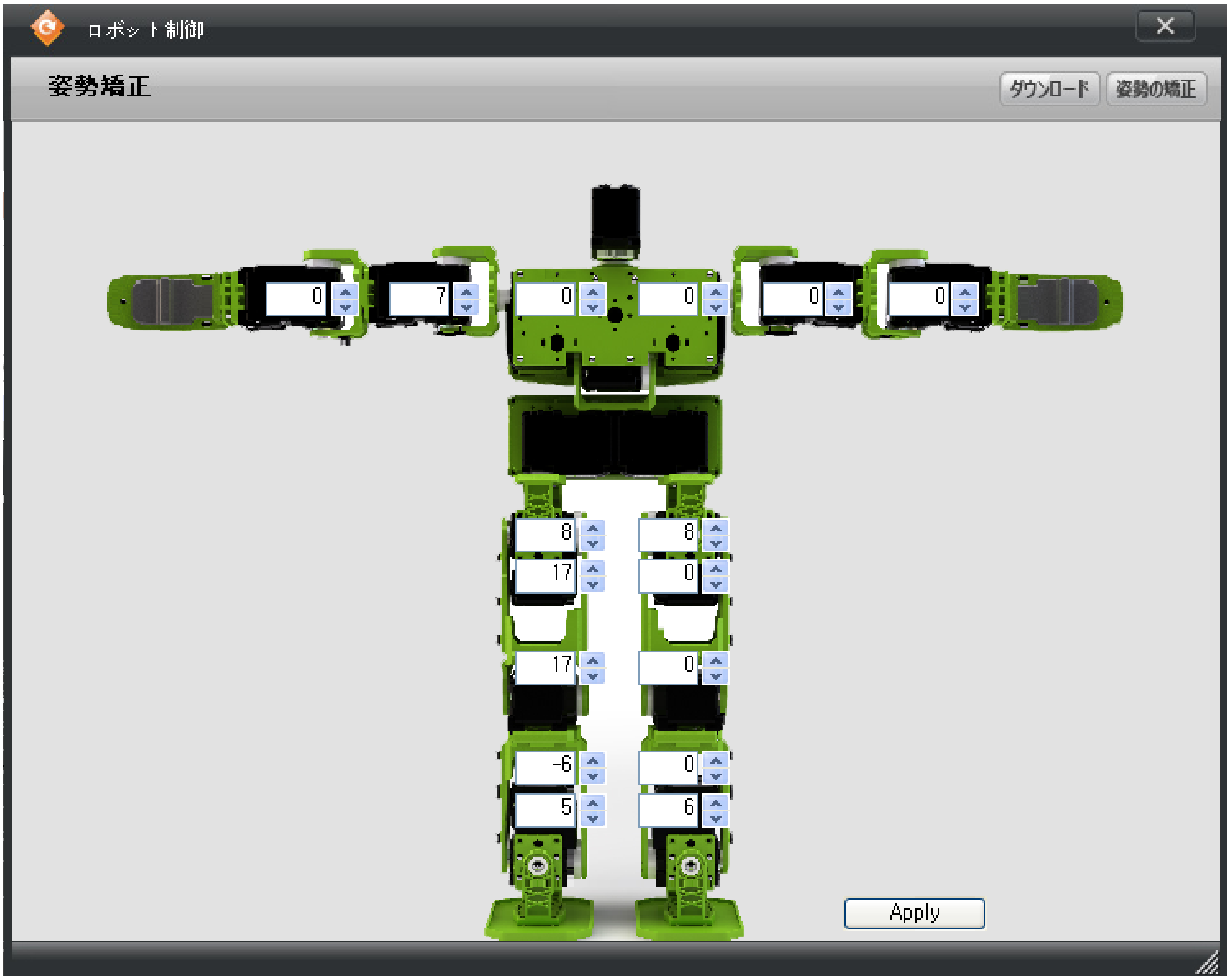
capture.dmt
hello.dmt
pushup.dmt
sitdown.dmt

[参考]
- 選択したフォルダを基準にモーションの情報を自動的に読み込みます。
- ダウンロードボタンをクリックすると、コントローラのモーションにモーションエディタのモーション情報が新たに書き込まれます。
- 聞くボタンは現在のコントローラに保存されているモーション情報を読み込みます。
- 実行ボタンをクリックすると、選択されたモーション情報がコントローラによって、ロボットで実行されます。
- モーションエディタとコントローラ間のデータ転送時にはロボットが自動的に再起動する可能性があります。


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
6
0
0
0
0
0
0
0
0
0
0
0
0
0
0
Apply

姿勢矯正をクリックすると、現在のロボットのキャリブレーション値が表示されます.
実際のロボットと比較しながら、各キャリブレーションの値を調整します.

キャリブレーションは127〜-128まで設定が可能です.上下のアイコンを使用して、数値を変更すると、実際のロボットが少しずつ動くことが確認できます。

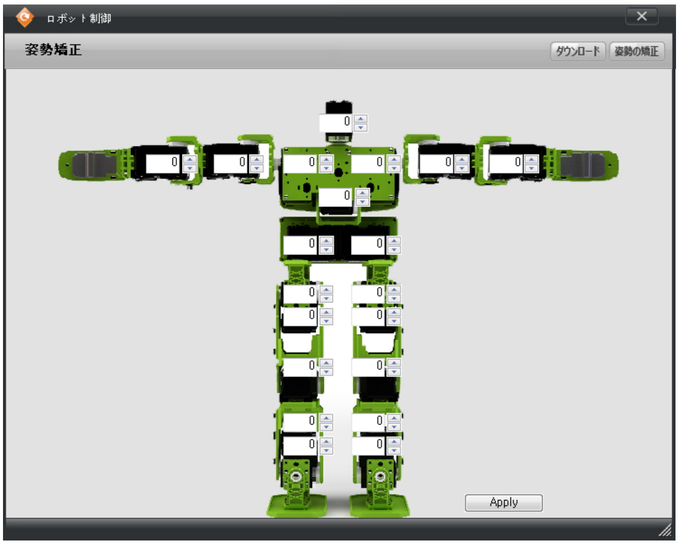

ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
7
0
0
0
0
8
8
17
0
17
0
-6
0
5
6
Apply

実際のロボットを見ながら動きを確認し、設定が正しければ、Applyをクリックします。
Applyをクリックするとロボットに保存されます。以降、ロボットを接続すると、調整された値の数字が表記されます。

キャリブレーション（ロボットの原点）

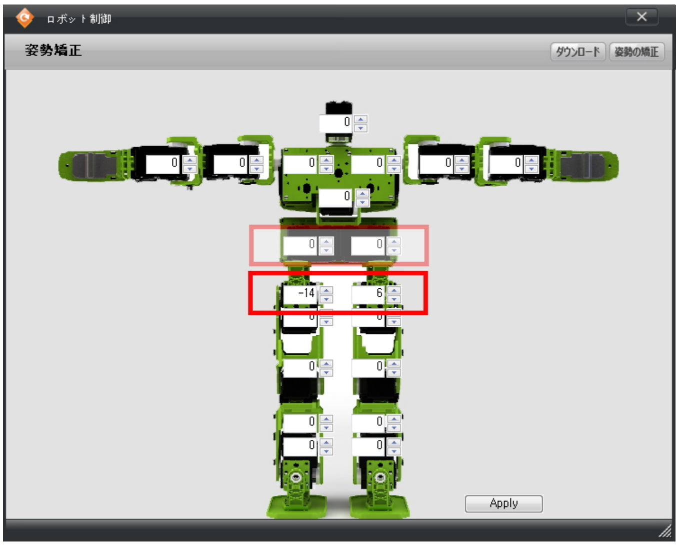
ロボットを組み立てた後、正確に組み立てられているかを確認して修正する作業です。もし、ロボットが正確に組み立てられていない状態で使用すると誤作動や望ましくない動作をする場合があります。

DR-SIMのロボットの制御をクリックして、実際のロボットモーターの正確な位置を変えながら調整します。


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
Apply

01

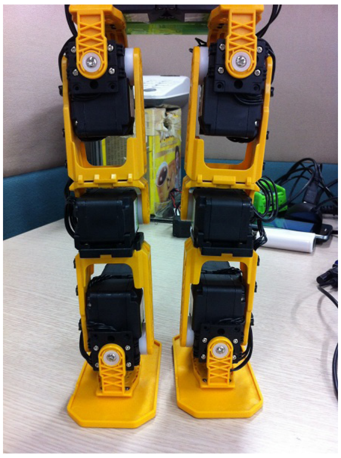
ロボット制御>姿勢矯正ボタンをクリックして、姿勢矯正ウィンドウが表示されたら、ロボットを軽く持ち上げて、組立状態を確認してください。


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
-14
6
Apply

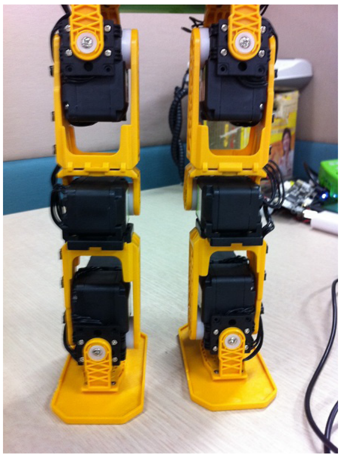
02

正面からロボットを見ながら、ロボットの両足のバランスを合わせます。

＊薄い四角ボックスは18DOF、20DOFロボットの場合に該当します。



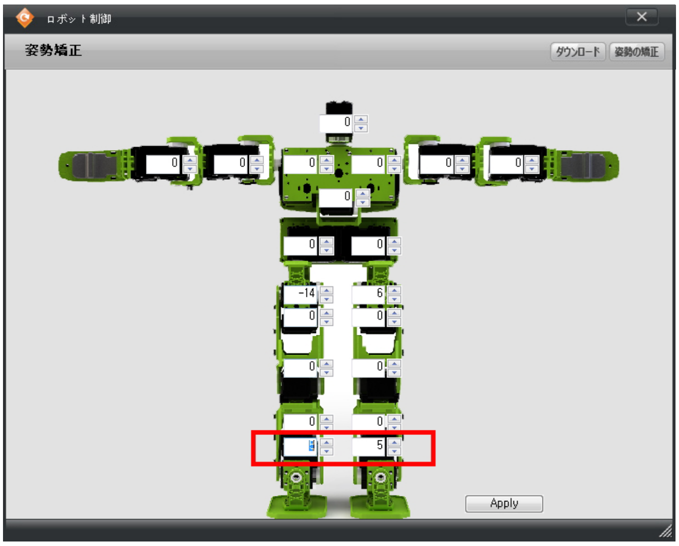
<設定前>



<設定後>

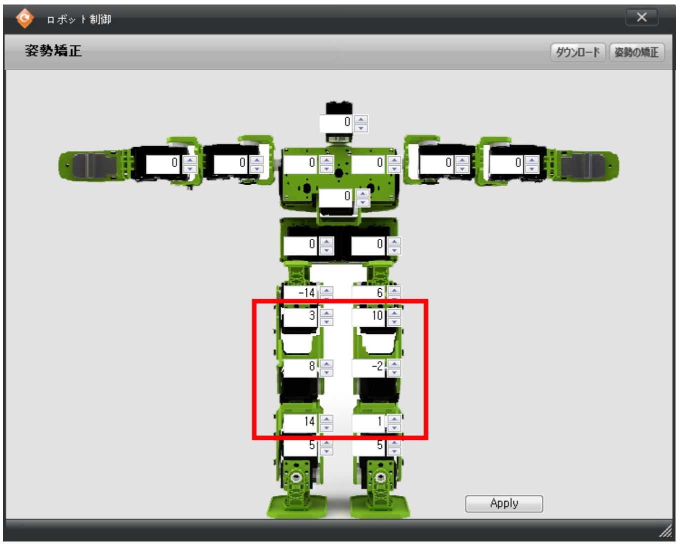
03

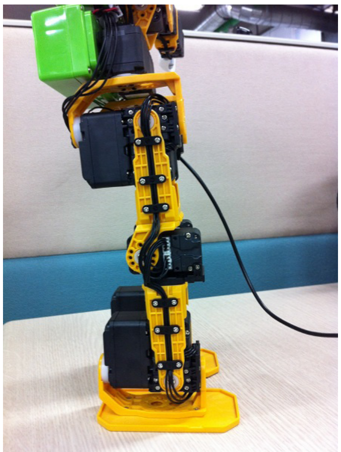
ロボットの両足が床に接地するように調整します.


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
0 0 0 0 0 0
0
0 0
-14 6
0 0
0 0
0 0
5
Apply

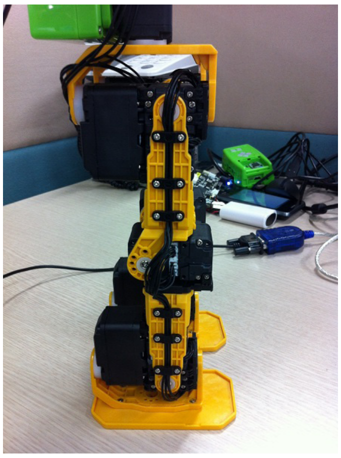
04

側でロボットを見ながら、各脚の垂直を合わせます.

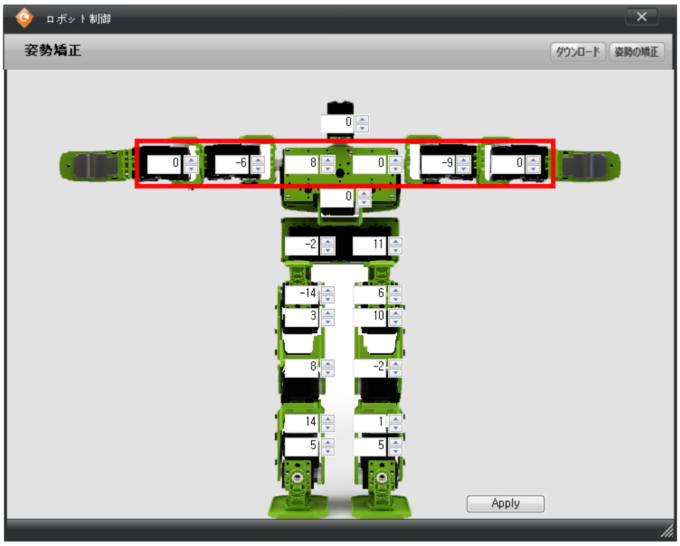

ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
0 0 0 0 0 0
0
0 0
-14 6
3 10
8 -2
14 1
5 5
Apply



<設定前>

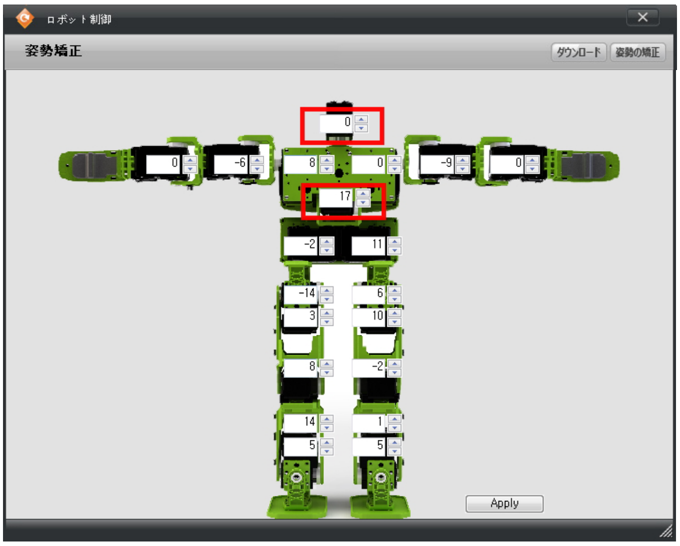


<設定後>


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
0
-6
8
0
-9
0
0
-2
11
-14
6
3
10
8
-2
14
1
5
5
Apply

05

上からロボットを見ながら、両腕が一になるよう調整します。


ロボット制御
姿勢矯正
ダウンロード
姿勢の矯正
0
0
-6
8
0
-9
0
17
-2
11
-14
6
3
10
8
-2
14
1
5
5
Apply

06

20DOFロボットの場合はロボットの腰と頭を調整します。

07

最後に部分的に値を調節してキャリブレーションを終了します。

有用な情報

モーターのIDを変更する

コントローラー(DRC)は、モーターのIDを使用して個々のモーターを識別しています。したがって、ロボットの組み立て時にモーターをID値に基づいて、適切な位置に組み立てることが重要です。

しかし、16軸のロボットを18軸または20軸のロボットに組み立てる場合やロボットの組み立て時にモーターを間違って組み立てた場合は、モーターのIDを変更する必要があります。

- 16軸から18軸または20軸に変更する場合は、再度組み立てる前に必ずモーターのIDを変更した後で、組み立ててください。
- モーターを違う位置に組み立てた場合、次のように変更します。

ex） 9番と10番のモーターの位置が変わった場合

9番 → 100番（20番から254番までのIDは使用していないIDです。）

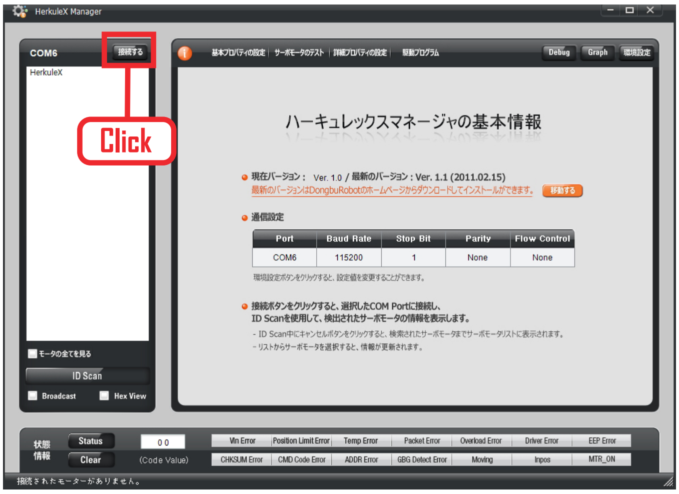
10番 → 9番

100番 → 10番

以下は、HerkuleX Managerプログラムを介して253番モーターのIDを15番に変更する例です。

（HerkuleX Managerは、Dongbu Robotのホームページからダウンロードできます。）

http://www.dongburobot.com/jsp/cms/view.jsp?code=100122


HerkuleX Manager
COM6
接続する
HerkuleX
Click
基本プロパティの設定
サーボモータのテスト
詳細プロパティの設定
駆動プログラム
Debug
Graph
環境設定
移動する
☐ モータの全てを見る
ID Scan
☐ Broadcast
☐ Hex View
状態情報
Status
Clear
0 0
(Code Value)
Vin Error
Position Limit Error
Temp Error
Packet Error
Overload Error
Driver Error
EEP Error
CHKSUM Error
CMD Code Error
ADDR Error
GBG Detect Error
Moving
Inpos
MTR_ON

ハーキュレックスマネージャの基本情報

- 現在バージョン： Ver. 1.0 / 最新のバージョン : Ver. 1.1 (2011.02.15)
  最新のバージョンはDongbuRobotのホームページからダウンロードしてインストールができます。
- 通信設定

Port	Baud Rate	Stop Bit	Parity	Flow Control
COM6	115200	1	None	None

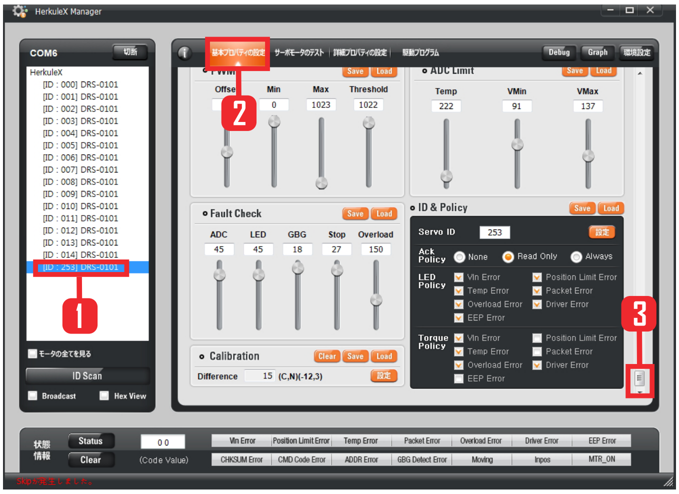
環境設定ボタンをクリックすると、設定値を変更することができます。

- 接続ボタンをクリックすると、選択したCOM Portに接続し、
  ID Scanを使用して、検出されたサーボモータの情報を表示します。
  - ID Scan中にキャンセルボタンをクリックすると、検索されたサーボモータまでサーボモータリストに表示されます。
  - リストからサーボモータを選択すると、情報が更新されます。

接続されたモーターがありません。

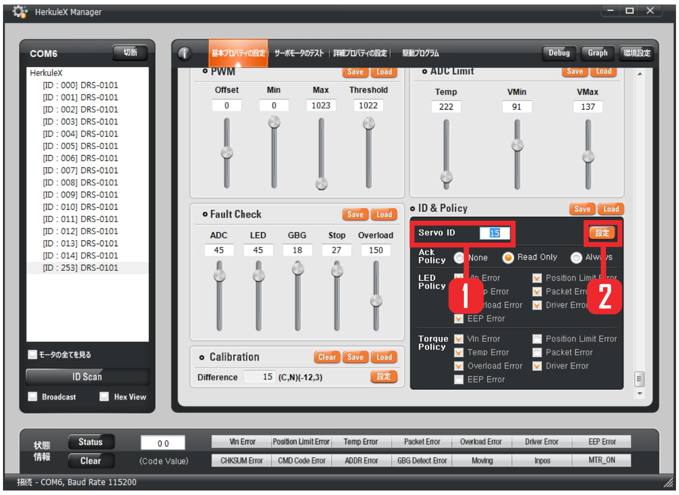
01

モーターをコントローラー（DRC）に接続し、HerkuleX Managerを実行します。COM Portの設定後、接続ボタンをクリックします。.



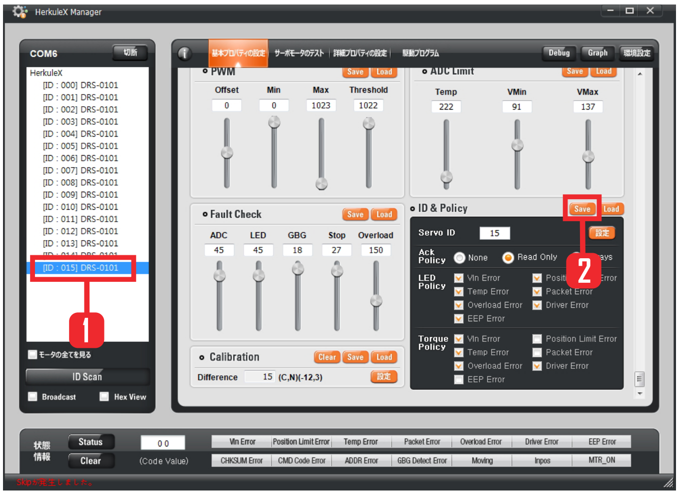
02

接続ボタンをクリックすると、コントローラー（DRC）に接続されたモーターが、左の画面に表示されます。　IDが253のモーターのIDを15に変更するために、[ID:253]　DRS-0101]をクリックします。基本プロパティー設定をクリックし、スクロールバーを利用して、IDとPolicyの画面が見えるように配置します。



03

ServoIDの項目に値を入力します。この例では、15を入力します。そして、設定をクリックします。設定をクリックすると、自動的にモーターのID　Scanが実行されます。



04

自動的にID　Scanが実行されると、モーターのIDが253から15に変更されます。この後、[ID:015]　DRS-0101をクリックし、Saveボタンをクリックして、モーターのIDを最終的に変更します。
（IDScan時に変更されたモーターのIDは、RAM　Register で変更された値です。RAM　RegisterのID値は、電源が供給されないと、消去されます。Saveボタンを利用して、変更されたモーターのIDをEEP Registerにも適用すれば、モーターのID変更が完了です。）

05

モーターの電源を切断した後、再接続します。その後、HerkuleX　ManagerでID　Scanを実行して、モーターのIDが正しく変更されていることを確認します。

お客様の権利

お客様は製品の使用中に故障が発生した場合、購入日より12ヶ月間無償サービスを受けることができる”大切な権利”を持っています。ただし、お客様の過失および自然災害によって故障が発生した場合は、無償サービス期間内でも有償処理になりますのでご了承ください。

有償サービス（お客様の費用負担）の責任

サービスお申し込み時に、次の場合には無償サービス期間内でも有償処理されます。

1. 故障ではない場合
·故障がない場合、サービスを要求すると、料金が発生しますので、必ず取扱説明書をお読みください。
（修理が不可能な場合は、別の基準に準じます。）

2. お客様の過失による故障の場合
·電源の異常や接続機器の不良で故障が発生した場合
·製品内に異物を投入したり、吸引して、故障が発生した場合
·本製品を落としたり、強い衝撃により製品が破損して、機能上の故障が発生した場合
·製品をお客様が任意に分解し、付属品が紛失または破損した場合
·シンナー、ベンジンなどの有機溶剤による変色や変形の場合
·ユーザーマニュアルの”安全のための注意事項”を守らなかったため、故障が発生した場合
·その他の製品自体の欠陥ではなく、外部の原因やお客様の過失により故障が発生した場合

3. 上記以外の場合
·自然災害（落雷、火災、風水害、ガス、塩害、地震など）によって故障が発生した場合
·電池のような通常の消耗部品を交換する場合

消費者被害補償の案内

消費者被害の種類		補償内容 保証期間内	補償内容 保証期間経過後
正常な使用状態で性能、機能上の故障が自然発生した場合（保証期間内）	ご購入10日以内に重要な修理を要する場合	製品交換または購入価格の払い戻し	
	購入1ヶ月以内に重要な部品の修理を要する場合	製品の返品	
	交換品が1ヶ月以内に重要な修理を要する場合	購入価格の払い戻し	
	交換不可能な時	無償修理	
	欠陥が発生した場合	製品交換または購入価格の払い戻し	
	同欠陥に対して、修理したが、故障の再発（4回目）時		
	いくつかの部位の故障で全4回の修理を受けたが、故障再発（5回目）時		
	修理不可能な時		
	修理用部品を保有していないため、修理が不可能な場合		定額の減価償却の金額に10%加算して返金
	消費者の修理を依頼した製品を事業者が紛失した場合		
	製品購入時の配送中に発生した損害		
消費者の故意、過失による性能、機能上の故障が発生した場合	修理が可能な場合	有償修理	
	修理用部品を保有していないため、修理が不可能な場合	製品交換または購入価格の払い戻し	

サービスの案内

Dongbu Robot

お客様は製品の使用中に故障が発生した場合、購入日より12ヶ月間無償サービスを受けることができる”大切な権利”を持っています。ただし、お客様の過失および自然災害によって故障が発生した場合は、無償サービス期間内でも有償処理になりますのでご了承ください。

✂ -

製品保証書

製品名 (Product Name) ：	
モデル名 (Model Name) ：	
製品番号 （Serial No.） ：	
購入日 (Purchase Date)：	年　　　月　　　日

お名前 ：	
住所 ：	
電話番号 (Tel. No.) ：	
電子メール （E-mail）	

※ 修理を依頼するときは、購入日が記載された本保証書の提示が必要な場合がありますので大切に保管してください。

✂ -

1. 本製品の品質保証は保証書に記載された内容で保証の恩恵を受けます。
2. 無償保証期間はお買い上げ日より算定されるため、購入日を記入してください。
（購入日の確認が出来なかった場合には、製造日から6ヶ月が経過した日から、品質保証期間を加算します。）
3. この保証書は再発行しませんので、使用説明書とともに大切に保管してください。

Dongbu Robot

www.hovis.co.kr/guide

www.dongburobot.com → English → Customer support → For Services → Archives

www.dongburobot.com → Quick Menu → For Service → Archives

Homepage : www.dongburobot.com